Fondamenti di Informatica

Marta Cialdea Mayer
Dipartimento di Informatica e Automazione
via della Vasca Navale 79
II piano

Roberto Maieli
Dipartimento di Informatica
Università di Roma "La Sapienza"
Via Salaria, 113 - III piano

Pagina del corso

`www.dia.uniroma3.it/~cialdea/teaching/FI-Com/`

Tutorato

ogni Venerdì dalle 18 alle 19, aula 17

Materiale didattico

- Questi lucidi (pagina del corso)
- M. Cialdea Mayer, C. Limongelli.
  Introduzione alla Programmazione Funzionale. Esculapio 2002.
- The Objective Caml system. Documentation and User's Guide.
- M. Cialdea Mayer, *Logica. Linguaggio, Ragionamento, Calcolo.* Esculapio 2002. Cap. 4

È disponibile un CD-ROM contenente il compilatore e il manuale del linguaggio, e un ambiente di sviluppo

LUG (Linux User Group) di Roma Tre
Via della Vasca Navale 79
I piano

- Compilatore di OCaml
- Manuale di OCaml
- XEmacs: un editor configurabile

Informatica

Informatica $\approx$ calcolatori ?

In inglese: *Computer Science*

- I calcolatori e le loro applicazioni poggiano in modo determinante sull'informatica
- Senza i calcolatori l'informatica sarebbe ancora agli albori.

INFORMATICA $\approx$ INFORmazione autoMATICA

Informatica: scienza delle informazioni, scienza che si occupa del trattamento automatico delle informazioni

Si è sviluppata dopo che l'invenzione dei calcolatori elettronici aveva consentito di realizzare la manipolazione di "simboli" in modo automatico.

Ma le radici dell'informatica sono remote:

- Tecniche di calcolo mediante meccanismi come l'abaco cinese: 2000 anni fa
- Algoritmi per eseguire le quattro operazioni sulla rappresentazione decimale dei numeri: VIII secolo d.C.
- La "pascalina", prima macchina per eseguire le quattro operazioni (Blaise Pascal): 1642

"Se ascolto dimentico, se vedo ricordo, se faccio capisco"
(antico proverbio cinese)

Per capire l'informatica, si deve **fare** qualcosa

Informatica: trattamento automatico delle informazioni per **risolvere problemi**

Fare cosa?

Metodologia dell'informatica:

1. Definizione del problema (che cosa si vuole fare?): **specifica** del problema, o **analisi dei requisiti**
2. Realizzazione della soluzione (come fare?):
   (a) Come rappresentare le informazioni coinvolte nel problema?
   (b) Quale procedimento risolutivo (algoritmo) adottare?
   (c) Infine, IMPLEMENTAZIONE della soluzione in un LINGUAGGIO DI PROGRAMMAZIONE
3. Verifica della soluzione ed eventuale ritorno alle fasi precedenti
   - Test su un insieme significativo di casi
   - Verifica formale (dimostrazione di proprietà importanti del programma)

Informazioni

Informazione: qualcosa che si possiede e che si può dare a un altro senza perderne il possesso

Se l'altro già possiede l'informazione, è come se la transazione andasse a vuoto: l'informazione presuppone un'incertezza da parte di colui che la riceve

Un'informazione serve per risolvere un'incertezza o per prendere decisioni: cioè per risolvere un **problema**.

Trattamento delle informazioni per risolvere problemi

- *Problema:* data una pianta di Roma, trovare un percorso per andare da un punto di origine a una destinazione

  Informazioni disponibili: pianta della città, origine, destinazione

  Informazione desiderata: percorso da origine a destinazione

  Calcolo automatico della soluzione: `atac.comune.roma.it`

- *Problema:* dato un elenco del telefono e il nome e cognome di una persona, trovare il suo numero di telefono

  Informazioni disponibili: elenco del telefono, nome e cognome

  Informazione desiderata: numero di telefono

- *Problema:* dato un elenco del telefono e un numero di telefono, trovare l'indirizzo corrispondente al numero di telefono

  Informazioni disponibili: elenco del telefono, numero di telefono

  Informazione desiderata: indirizzo

Sull'elenco di Roma occorrerebbero circa 13 giorni!

Importanza del modo in cui è strutturata l'informazione (Strutture di dati)

Oggetti

L'informazione viene rappresentata mediante mediante

oggetti simbolici: DATI

Oggetti semplici:

- Caratteri alfabetici o numerici
- Nomi, parole: successioni di caratteri alfabetici (*stringhe*)
- Numeri:
  - Numeri naturali: successioni di caratteri numerici
  - Numeri interi: numeri naturali con segno
  - Numeri reali: successioni di caratteri numerici, seguiti dal punto decimale, seguito da una successione di caratteri numerici
- Booleani (o valori di verità): Vero e Falso

Oggetti complessi o strutturati:

- Coppie di oggetti (semplici o strutturati)
- Triple di oggetti
- Sequenze di oggetti
- Strutture complesse, come ad esempio un albero genealogico

Problemi

- Dati due numeri, trovarne il maggiore
- Dato un numero reale, calcolarne la radice quadrata
- Data un'equazione di secondo grado, trovarne le soluzioni

Un problema specifica una relazione tra

DATI DI INGRESSO (INPUT) e **DATI DI USCITA (OUTPUT)**

Trattamento automatico delle informazioni

Soluzione di un problema mediante l'esecuzione di un **procedimento automatico**: una sequenza di azioni elementari (ISTRUZIONI) su oggetti (DATI)

Le istruzioni devono essere

- Non ambigue

Aggiungere sale e pepe quanto basta: NO!

- Eseguibili

Aggiungere 40 g. di coda di brontosauro: NO!

La descrizione del procedimento deve essere finita (anche se la sua esecuzione potrebbe non terminare mai).

Procedimento automatico: Algoritmo

"Algoritmo" deriva dal nome del grande matematico persiano del IX secolo, Mohamed Ben-Musa Al-Khuvarizmi, che per primo ha descritto in modo formale e generale le regole per eseguire le operazioni sulle rappresentazioni decimali dei numeri interi

Un algoritmo è

- un metodo generale o sistematico per risolvere un insieme di problemi
- un metodo "sicuro" o imparziale, che può essere eseguito da una macchina

Un ELABORATORE è proprio una macchina in grado di risolvere problemi mediante l'esecuzione di algoritmi

Perché un elaboratore possa eseguire un procedimento, questo deve essere descritto in un linguaggio comprensibile dall'elaboratore stesso:

LINGUAGGIO DI PROGRAMMAZIONE

Esempio: un algoritmo di confronto di sequenze di caratteri

PROBLEMA: date due sequenze di caratteri S1 e S2, decidere se sono uguali o no

INPUT: S1, S2 (stringhe)
OUTPUT: Vero o Falso (booleani)

ALGORITMO:

```
CONFRONTA (S1,S2):
   SE S1 e S2 hanno lunghezza differente,
        ALLORA termina riportando valore FALSO
   ALTRIMENTI:
      sia L la lunghezza di S1 e S2
      PER OGNI n compreso tra 1 e L:
         SE l'n-esimo carattere di S1 e' diverso dall'n-esimo
            carattere di S2
         ALLORA termina riportando valore FALSO
         ALTRIMENTI passa al valore successivo di n
      Una volta esauriti i valori di n,
      termina riportando valore VERO
```

L'algoritmo CONFRONTA è generale: vale per stringhe qualsiasi

Verso un'implementazione dell'algoritmo COFRONTA

Utilizziamo le seguenti notazioni:

`S.[n]` per indicare l'n-esimo carattere della stringa `S`

`length(S)` per indicare la lunghezza della stringa `S`

`IF ... THEN ... ELSE` al posto di SE ... ALLORA ... ALTRIMENTI

`FOR n=1 TO L` : PER OGNI n compreso tra 1 e L

`RETURN ...` : termina, riportando il valore ...

`TRUE, FALSE` : vero e falso

`<>` : diverso

```
CONFRONTA (S1,S2):
   IF length(S1) <> length(S2) THEN RETURN FALSE
   ELSE FOR n=1 to length(S1):
           IF S1.[n] <> S2.[n]
           THEN RETURN FALSE
        RETURN TRUE
```

Non esiste un solo algoritmo per risolvere un dato problema

```
CONFRONTA-R (S1,S2):
   IF length(S1) = length(S2) THEN RETURN FALSE
   ELSE RETURN CICLO (S1,S2,length(S1))

CICLO (S1,S2,n):
   IF n=0 THEN RETURN TRUE
   ELSE IF S1.[n] <> S2.[n]
        THEN RETURN FALSE
        ELSE RETURN CICLO (S1,S2,n-1)
```

`CICLO` risolve un **sottoproblema** utile per risolvere il problema del confronto:

date due stringhe S1 e S2 della stessa lunghezza,
e un intero n (compreso tra 1 e la lunghezza delle stringhe),
verifica se i caratteri delle stringhe in posizione 1,2,...,n sono uguali

Esempio di "esecuzione" dell'algoritmo CICLO

```
CICLO (S1,S2,n):
   IF n=0 THEN RETURN TRUE
   ELSE IF S1.[n] <> S2.[n]
        THEN RETURN FALSE
        ELSE RETURN CICLO (S1,S2,n-1)
```

```
CICLO("pippo","piano",2):
   2 non e' uguale a 0, quindi continua
   "pippo".[2]="piano".[2], quindi riporta:

        CICLO("pippo","piano",1):
           1 non e' uguale a 0, quindi continua
           "pippo".[1]="piano".[1], quindi riporta:

                CICLO("pippo","piano",0):
                0=0, quindi riporta TRUE
```

Linguaggi di programmazione

Esistono diverse tipologie di linguaggi di programmazione

Linguaggi di basso livello Le istruzioni del linguaggio sono molto semplici, corrispondono alle istruzioni eseguibili direttamente dall'elaboratore

Linguaggi di alto livello Il linguaggio prevede istruzioni di più alto livello, che, per essere eseguite dall'elaboratore, saranno tradotte (COMPILATE o INTERPRETATE) in programmi di basso livello

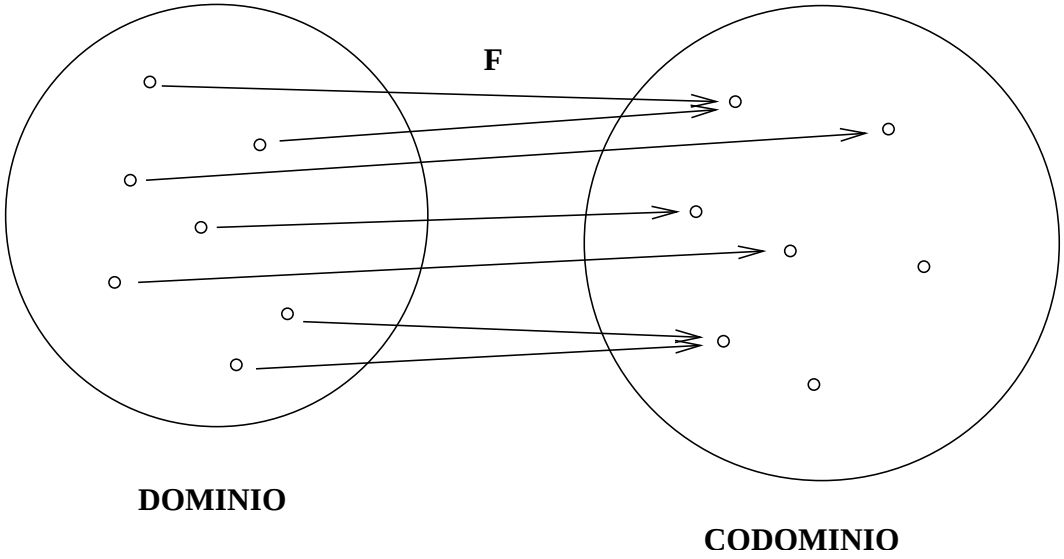
Tra i linguaggi di alto livello:

Linguaggi di programmazione funzionale

Un programma è un'operazione che associa un input con un output

UN PROGRAMMA È UNA FUNZIONE

FUNZIONI



F associa a ogni elemento del DOMINIO un elemento del CODOMINIO

Il **TIPO** di **F** è: DOMINIO $\rightarrow$ CODOMINIO

COS'È UN TIPO?

Un tipo è un insieme di oggetti (**VALORI**).

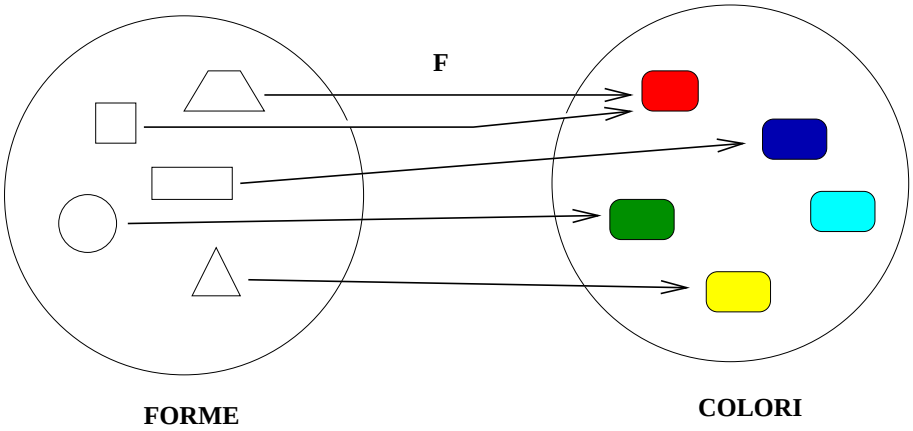
Se A è un tipo e $x \in A$, diciamo che x è di tipo A

$$x : A$$

Ad esempio: $3 : \mathbb{N}$ $(2, 5) : \mathbb{N} \times \mathbb{N}$ $(1, 2, 3) : \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, o anche: $(1, 2, 3) : \mathbb{N}^3$

Il tipo $A \to B$ è l'insieme di tutte le funzioni che hanno A come dominio e B come codominio.

ESEMPIO 1



$$F : FORME \rightarrow COLORI$$

F si applica a un elemento di FORME (**ARGOMENTO DELLA FUNZIONE**)
e **RIPORTA** un elemento di COLORI come **VALORE**

ESEMPIO 2

Se $square$ è la funzione che associa a ogni numero naturale

$$n \mapsto n^2$$

$$square : \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{lll} square(0) = 0 & square(1) = 1 & square(2) = 4 \\ square(3) = 9 & square(4) = 16 & square(5) = 25 \\ & \ldots & \end{array}$$

$square$ è quella funzione che, applicata a n, riporta n^2

$$square = function\ n \to n^2$$

FUNZIONI A PIÙ ARGOMENTI

Sia *times* la funzione che associa

$$(n, m) \mapsto n \times m$$

per ogni $n, m \in \mathbb{N}$.

$$times = function\ (n, m) \to n \times m$$

(n, m) è una **coppia** di numeri

Qual è il tipo di *times*?

Il suo codominio è $\mathbb{N}$

Il suo dominio è l'insieme $\{(n, m) \mid n, m \in \mathbb{N}\}$

Prodotto cartesiano di $\mathbb{N}$ per se stesso:

$$\mathbb{N} \times \mathbb{N}$$

PRODOTTO CARTESIANO

Se $A = \{0, 1, 2\}$ e $B = \{rosso, verde\}$,

$$\begin{aligned} A \times B \ = \ & \{(0, rosso), (0, verde), (1, rosso), (1, verde), \\ & (2, rosso), (2, verde)\} \end{aligned}$$

Quindi:

$$\begin{aligned} \mathbb{N} \times \mathbb{N} \ = \ & \{(0,0), (0,1), (0,2), (0,3), ... \\ & (1,0), (1,1), (1,2), (1,3), ... \\ & (2,0), (2,1), (2,2), (2,3), ... \\ & .....\} \end{aligned}$$

e

$$sum : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

Quando si applica sum a (n, m), diciamo che n è il primo argomento, m è il secondo argomento.

Funzioni a n argomenti

Se una funzione si applica a n argomenti, appartenenti, rispettivamente agli insiemi A_1,...,A_n e riporta un valore nell'insieme B, il suo tipo è

$$A_1 \times A_2 \times ... \times A_n \to B$$

Il suo dominio è un insieme di tuple di n elementi:

$$\{...., (a_1, a_2, ..., a_n), ...\}$$

FUNZIONI CHE RIPORTANO COPPIE COME VALORI

Sia *quorem* la funzione che

applicata a due numeri naturali n e m

riporta

il quoziente intero (n/m) e il resto $(n\ mod\ m)$ della divisione di n per m

$$quorem = function(n, m) \rightarrow (n/m, n\ mod\ m)$$

$$quorem(3,2) = (1,1) \qquad quorem(3,3) = (1,0)$$
$$quorem(7,2) = (3,1) \qquad quorem(15,6) = (2,3)$$

$$quorem : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$$

ATTENZIONE: qual è il valore di $quorem(1,0)$?

quorem non è definita per l'argomento $(1,0)$

Funzioni totali e parziali

FUNZIONI TOTALI: sono definite per ogni elemento del dominio

FUNZIONI PARZIALI: possono essere indefinite per alcuni elementi del dominio

Una funzione parziale è ovviamente totale se si restringe opportunamente il suo dominio.

$$quorem' : \mathbb{N} \times (\mathbb{N} - \{0\}) \to \mathbb{N} \times \mathbb{N}$$

è totale

FUNZIONI POLIMORFE

Consideriamo la funzione $first$, tale che

$$first(x, y) = x$$

$$first = function(x, y) \rightarrow x$$

x e y potrebbero essere di qualunque tipo.

$first(0, 1) = 0$	$first(quadrato, rosso) = quadrato$
$first(0, rosso) = 0$	$first(\pi, 0) = \pi$
$first(\pi, 1.5) = \pi$	...

$first$ è di tipo

$\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

$FORME \times COLORI \rightarrow FORME$

$\mathbb{N} \times COLORI \rightarrow \mathbb{N}$

$\mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}$

$\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

....

$first$ ha molti tipi, tutti quelli della forma:

$$TIPO_1 \times TIPO_2 \rightarrow TIPO_1$$

$first$ è una funzione **POLIMORFA**.

Possiamo identificare il suo **TIPO PIÙ GENERALE** utilizzando

VARIABILI DI TIPO

$$first : \alpha \times \beta \rightarrow \alpha$$

Ogni tipo di $first$ è un'**ISTANZA** del suo tipo più generale.

APPLICAZIONE di funzioni

$$F(x) = y$$

x è l'argomento della funzione
y è il valore dell'applicazione $F(x)$

Se

$F : A \to B$

$x : A \quad$ (cioè $x \in A$)

allora

$$F(x) : B \quad \text{(cioè } F(x) \in B\text{)}$$

IL CALCOLO COME RIDUZIONE

Calcolare significa ridurre un'espressione a un **VALORE**
Un valore è un'espressione non ulteriormente riducibile

Un passo di riduzione

Consideriamo una funzione:

$$f = function\ x \to E$$

- x è il **parametro formale** della funzione
- E è il **corpo** della funzione

Quando si applica la funzione a un argomento:

$$f(M) \qquad \text{(o semplicemente } f\ M)$$

- M è il **parametro attuale** della funzione

l'espressione $f(M)$ si semplifica (si avvicina a un valore) in

$$E[M/x]$$

che è l'espressione che si ottiene da E sostituendo tutte le occorrenze del parametro formale x con il parametro attuale M
Ad esempio:

$$(function\ n \to sum(3, n))\ 5 \mapsto sum(3, 5)$$

Calcolo del valore di un'espressione mediante una sequenza di riduzioni: esempi

$square\ 5$
$\mapsto\ (function\ n \to n^2)5$
$\mapsto\ 5^2$
$\mapsto\ 25$ valore di $square\ 5$

$first(15, 20)$
$\mapsto\ (function(n, m) \to n)(15, 20)$
$\mapsto\ 15$ valore di $first(15, 20)$

$quorem(square(5), first(15, 20))$
$\mapsto\ quorem((function\ n \to n^2)5, (function(n, m) \to n)(15, 20))$
$\mapsto\ ...$
$\mapsto\ quorem(25, 15)$
$\mapsto\ (function(n, m) \to (n/m, n\ mod\ m))(25, 15)$
$\mapsto\ (25/15, 25\ mod\ 15)$
$\mapsto\ (1, 10)$ valore di $quorem(square(5), first(15, 20))$

Esercizi

1. Sia $F : \mathbb{N} \to \mathbb{N}$ la funzione che, applicata a $n > 0$, riporta $n - 1$:

$$F = function\ n \to n - 1$$

   F è una funzione totale o parziale?

2. Sia $minus$ la funzione di sottrazione sui numeri interi $\mathbb{Z}$. Scrivere il tipo di $minus$.

3. Sia $A = \{0, 1, 2\}$ e $B = \{\text{“}pippo''\text{,''}\ pluto''\text{,''}\ paperino''\}$. Scrivere il prodotto cartesiano $A \times B$ e il prodotto cartesiano $B \times A$.

4. Scrivere un'espressione $function ... \to ...$ che denoti la funzione $second$ che, applicata a una coppia, riporti il secondo elemento della coppia. Ha un unico tipo? Qual è il suo tipo più generale? Calcolare il valore dell'espressione $second(sum(3, 5), square\ 2)$, mediante una sequenza di riduzioni.

5. Sia $sommadue = function\ n \to n + 2$ di tipo $\mathbb{N} \to \mathbb{N}$. Calcolare il valore dell'espressione

$$sommadue(square\ 4)$$

6. Si consideri la funzione max, che applicata a una coppia di naturali (n, m), riporta il maggiore tra i due (come caso particolare, $max(n, n) = n$). Qual è il tipo di max?

7. Un predicato è una funzione particolare. Come si può caratterizzare?