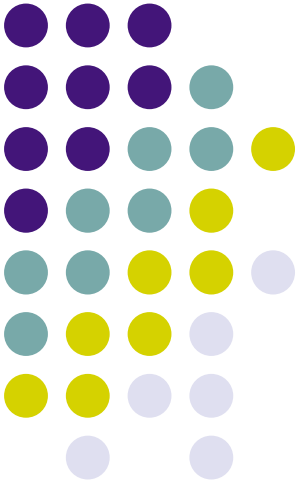
Programmazione
Modulo 1

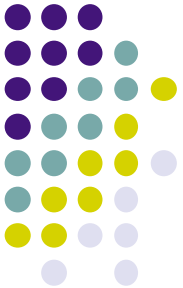
A.A. 2012/13

Introduzione

Docente: Dott.ssa Sabina Rossi

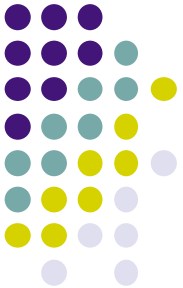


Obiettivi



- Il corso si propone di introdurre gli elementi base della programmazione per affrontare i problemi che ammettono soluzioni algoritmiche.
- Al termine del corso lo studente dovrà saper progettare e formalizzare semplici programmi, sviluppati secondo i paradigmi funzionale e imperativo.
- Dovrà inoltre essere in grado di analizzare la struttura logica di un programma per verificarne la correttezza.

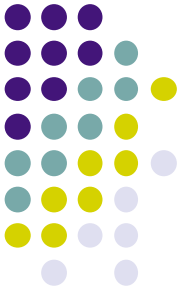
Programmazione (12 cfu)



Il corso è diviso in due moduli:

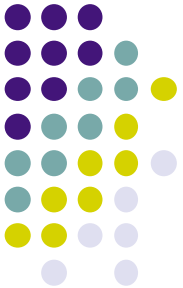
- *Primo semestre*
  - **Modulo 1** = 48 ore = 6 CFU
    - Dott.ssa Sabina Rossi
- *Secondo semestre*
  - **Modulo 2** = 48 ore = 6 CFU
    - Dott. Andrea Marin

Testi di riferimento



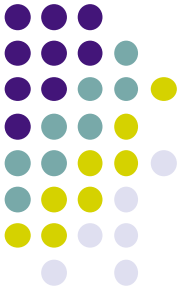
- Modulo 1
  - M. Hailperin, B. Kaiser, K. Knight : Concrete Abstractions Brooks/Cole Publishing Company, 1999 http://gustavus.edu/+max/concrete-abstractions-pdfs/index.html
  - Dispense: traduzione in italiano e con il linguaggio ML
- Modulo 2
  - A. Kelley, I. Pohl. C didattica e programmazione. E. Pearson-Addison Wesley,2004
  - B. Kernighan, D. Ritchie. Il Linguaggio C. Ed. Pearson-Prentice Hall, 2004.

Modalità di esame



- Prova scritta
- Prova orale

Prova scritta mediante prove intermedie

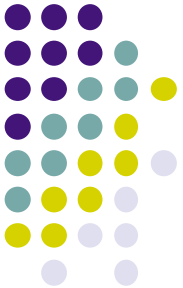


- Primo semestre
  - Prova scritta
  - Prova scritta

- Secondo semestre
  - Prova pratica in laboratorio
  - Prova scritta

- Il voto della prova scritta si calcola facendo la media delle quattro prove intermedie.

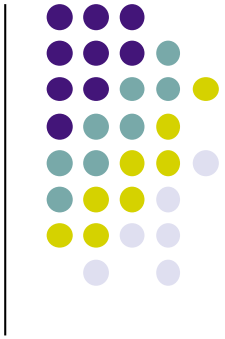
Prova orale mediante esercitazioni in laboratorio



- <u>Esercitazioni settimanali da svolgere in laboratorio</u>

- Coloro che consegnano almeno il 70% delle esercitazioni con valutazione sufficiente sono esonerati dall'orale

Pagina web del corso

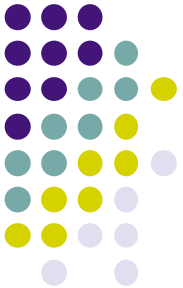


http://moodle.dsi.unive.it/

Create new account

- News
- Forum
- Dispensa
- Esercizi
- …

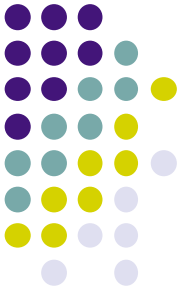
Orario modulo 1



- Lezioni in aula 1
  - Mercoledì 10.30-12.00
  - Giovedì 10.30-12.00

- Laboratorio
  - Mercoledì 15.45-17.15 (matricole pari)
  - Giovedì 15.45-17.15 (matricole dispari)

Ricevimento



Dott.ssa Sabina Rossi

- Mercoledì dalle 14.00 alle 15.00
- oppure su appuntamento scrivendo a

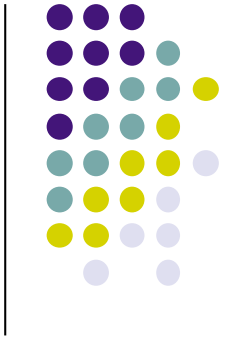
srossi@dais.unive.it

Dott. Andrea Marin

- Giovedì dalle 14.30 alle 15.30
- oppure su appuntamento scrivendo a

marin@dais.unive.it

Laboratorio



Da Ottobre:

- Esercitazioni in laboratorio

Tutors:

- Mohamed Abbadi
- Enrico Steffinlongo
- Alvise Spanò
- Mauro Tempesta

Che cos'è un linguaggio di programmazione ?

- Un linguaggio di programmazione è un mezzo per poter scrivere "programmi", allo scopo di controllare il computer, facendogli eseguire determinati compiti.

Quanti linguaggi esistono ?

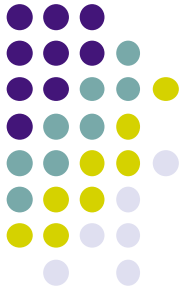
- Così come le persone possono comunicare usando tante lingue diverse, anche i computer possono essere istruiti mediante tanti linguaggi diversi.

La Babele dei linguaggi...

ASSEMBLER, FORTRAN, ALGOL, COBOL, BASIC, PASCAL, C, C++, LISP, ADA, SMALLTALK, LOGO, JAVA, ASP, PHP, PERL, DELPHI, DBASE, MODULA, MATLAB, SQL, MUMPS, PYTHON, OBERON, OCTAVE, EIFFEL, PARADOX, ecc.

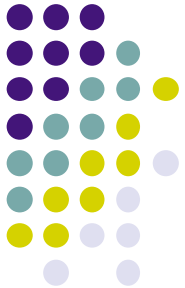
Pieter Bruegel, *La Torre di Babele*, 1563

Paradigmi di Programmazione



- *Paradigmi di tipo procedurale:*
  come risolvere il problema
  - *Paradigma imperativo*
    - Basato sul concetto di variabile e di stato, consiste nel decidere la migliore trasformazione sequenziale dei valori delle variabili che porti al risultato desiderato.
  - *Paradigma ad oggetti*
    - Un programma in un linguaggio di questo modello consiste di oggetti che inviano messaggi a ciascun altro. Questi oggetti corrispondono in genere agli oggetti presenti nello spazio del problema, come persone, macchine, dipartimenti, … .

Paradigmi di Programmazione

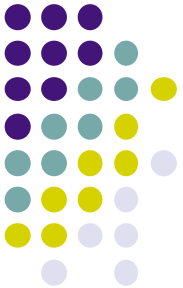


- *Paradigmi di tipo dichiarativi:*

  cosa si vuole risolvere

  - *Paradigma logico*
    - Si basa sull'idea di considerare i predicati logici del primo ordine (ristretti ad una certa forma sintattica) come delle *specifiche eseguibili*.
  - *Paradigma funzionale*
    - Si concentra sul problema, fornendo una descrizione astratta delle funzioni che devono essere utilizzate per risolvere il problema.

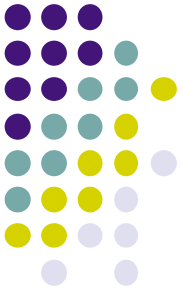
Quale linguaggio scegliere ?



- Ogni linguaggio è stato creato e ottimizzato per certi tipi di applicazioni.

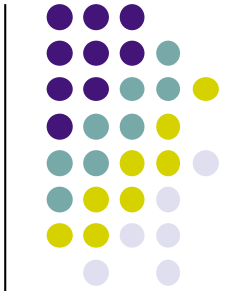
In sintesi:

- **Applicazioni Matematiche**: Fortran, Algol, Matlab, Lisp, Caml, C++
- **Sistemi operativi:** Assembler, C, C++
- **Programmi gestionali:** Visual Basic, Cobol, C, C++
- **Programmi multipiattaforma:** Java
- **Programmi didattici:** Pascal, C, C++
- **Giochi**: C++, Assembler, Java, Flash



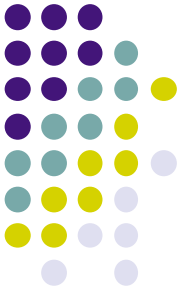
I linguaggi che studierete nei corsi di Programmazione

ML



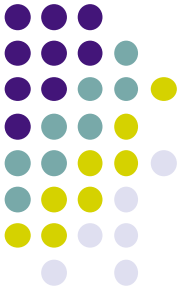
- ML (Meta Language)
  - Nato alla fine degli anni ‘70 come linguaggio di specifica per strategie di prove
  - Si basa sul concetto di funzione
  - Inferenza e controllo dei tipi

CaML e OCaML



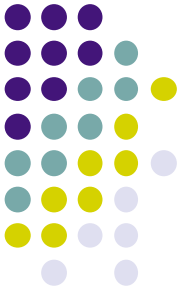
- CaML (Categorical Abstract Machine Language) è una implementazione di ML sviluppata in Francia presso l'INRIA e l'École Normale Supérieure (ENS) nel 1984.
- Ocaml (Objective CaML), creato nel 1996, fornisce una estensione, orientata ad oggetti (e con un frammento imperativo), di CaML.

F#



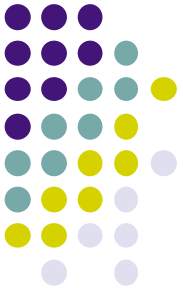
- Creato da Don Syme di Microsoft Research come variante Ocaml.
- Di recente incluso nella famiglia di prodotti Microsoft® .NET Framework.
- F# garantisce indipendenza dai tipi, prestazioni elevate e la capacità di fungere da linguaggio di script, il tutto come parte integrante dell'ambiente .NET.

C



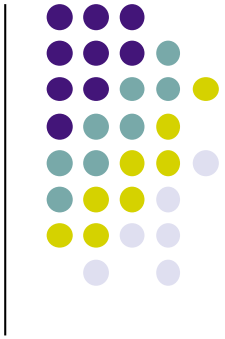
- Il C fu ideato nel 1972 da da Kernighan e Ritchie per implementare il sistema operativo Unix.
- L'idea da cui nasce è la gestione dei dati a basso livello e l'efficienza.
- Linguaggio compatto, non molto leggibile.

C++



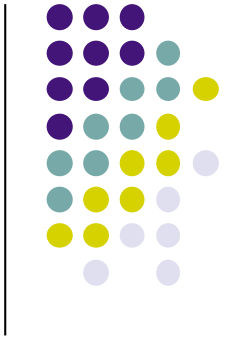
- Progettato da Bjarne Stroustrup nel 1983, risulta un aggiornamento del C, con l'aggiunta di elementi che supportano la programmazione ad oggetti.
- Il suo principale contributo è stato quello di aver fatto "coesistere" il paradigma imperativo e quello ad oggetti in un unico linguaggio.

JAVA



- Progettato da James Gosling, deve gran parte del suo successo soprattutto all'avvento del *World Wide Web* nel 1993.
- Il suo stile è basato su quello del C++ e contiene molte utitilies dei linguaggi imperativi e ad oggetti.
- Indipendente dall'architettura: il compilatore non crea degli applicativi eseguibili dal computer, ma dei file .class che devono essere interpretati dalla JVM (Java Virtual Machine, un microprocessore virtuale) implementata sui vari sistemi.
- Presenta un'ampia gerarchia di classi *built-in* con molti strumenti, specialmente grafici.
- È in grado di supportare il *multi-threaded processing* per le esecuzioni concorrenti.

CaML e OCaML



- Il sito di riferimento è:

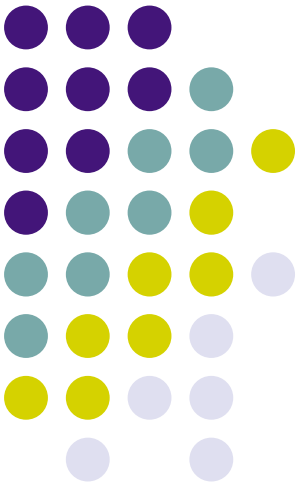
  http://caml.inria.fr

  da cui è possibile scaricare distribuzioni di Caml e Ocaml per Linux, MacOS e Windows.

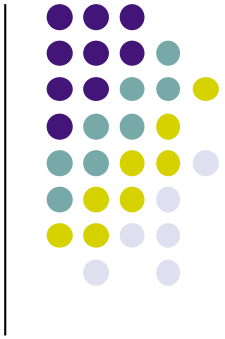
- Il manuale on line al sito

  http://caml.inria.fr/ocaml/htmlman/index.html

Programmazione imperativa e funzionale

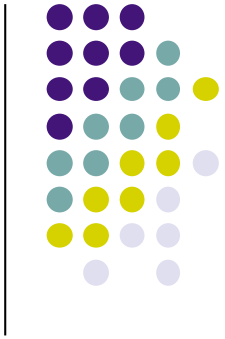


La programmazione imperativa



- Il modello di calcolo è basato sull'hardware (architettura di Von Neumann)
- Elementi di base nei linguaggi imperativi:
  - variabili (imitano celle di memoria)
  - assegnazione (costrutto primitivo)
- Programmare: pianificare il flusso di informazioni
  - La programmazione imperativa è basata su COMANDI, che operano sulla MEMORIA (stato del programma).
  - Per capire il programma (e provarne la correttezza) occorre tenere traccia delle modifiche dello stato.

La programmazione imperativa



Esempio : calcolo del fattoriale di un numero

$$Fact(n) = 1 \times 2 \times ... \times n - 1 \times n = n!$$

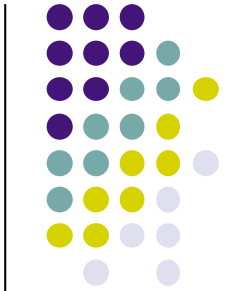
- ALGORITMO per calcolare Fact(n):

  Inizializzare f =1
  Finché n > 1, esegui le seguenti operazioni:
  f ← f × n
  n ← n − 1
  Restituisci f

- PROGRAMMA

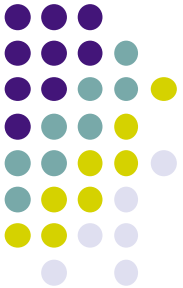
```
int fact (int n) {
int f = 1;
while(n>1){
f=f*n;
n=n-1; }
return f;
}
```

Linguaggi dichiarativi



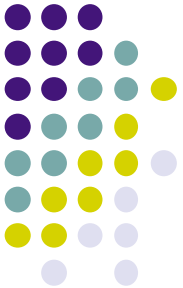
- Un programma è più vicino alla descrizione di che cosa si deve calcolare, piuttosto che a come calcolare.
  - Linguaggi di programmazione logica
  - Linguaggi di programmazione funzionale

La programmazione funzionale



- La programmazione funzionale richiede uno stile di pensiero decisamente astratto, fortemente ispirato ai principi della matematica.
- Il codice di un programma funzionale è solitamente più conciso del corrispondente codice procedurale.
- Un programma è una collezione di dichiarazioni di funzioni, che possono richiamarsi l'una con l'altra.
- Le funzioni possono costituire l'argomento o il valore di altre funzioni.
- I linguaggi funzionali consentono quindi l'uso di funzioni di ordine superiore, cioè funzioni che prendono funzioni come argomento o riportano funzioni come valore.

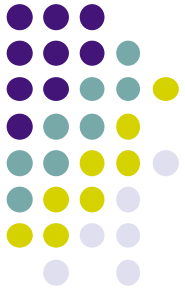
La programmazione funzionale



- Non esistono strutture di controllo predefinite per la realizzazione di cicli for o while: il principale meccanismo di controllo è la *ricorsione*.

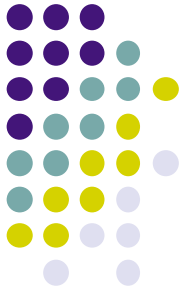
- La programmazione funzionale si basa sul λ-calcolo (lambda-calcolo), sviluppato per analizzare formalmente le definizioni di funzioni, le loro applicazioni e i fenomeni di ricorsione.

La programmazione funzionale



- Un programma è un'operazione che associa un input con un output: un programma è una *funzione*.
- Strutture di controllo:
  - applicazione di una funzione a un argomento
  - composizione di funzioni
  - ricorsione
- I costrutti di base sono ESPRESSIONI, non comandi.
- La programmazione funzionale è quindi una *programmazione orientata alle espressioni*, nella quale la modalità fondamentale di calcolo è la "valutazione di espressioni", che consiste nel calcolare il valore di una data espressione, semplificando la fin dove possibile.

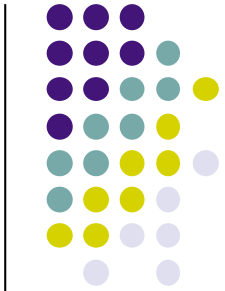
La programmazione funzionale



- Si calcola riducendo un'espressione a un'altra più semplice, fino a ottenere un VALORE (un'espressione non ulteriormente semplificabile)
  - Esempio: (6 + 3) × (8 − 2) ← 9 × (8 − 2) ← 9 × 6 ← 54

- **VALUTAZIONE**
  - Le espressioni hanno un valore
  - Non ci sono effetti collaterali
    - Esempio: (6 + 3) × (8 − 2) viene valutato, ma il valore non è "messo" da nessuna parte

La programmazione funzionale



- **Esempio**: Calcolo del fattoriale di un numero

$$\text{Fact}(n) = \begin{cases} 1 \text{ se } n = 0 \\ n * \text{fact}(n-1) \text{ se } n > 0 \end{cases}$$

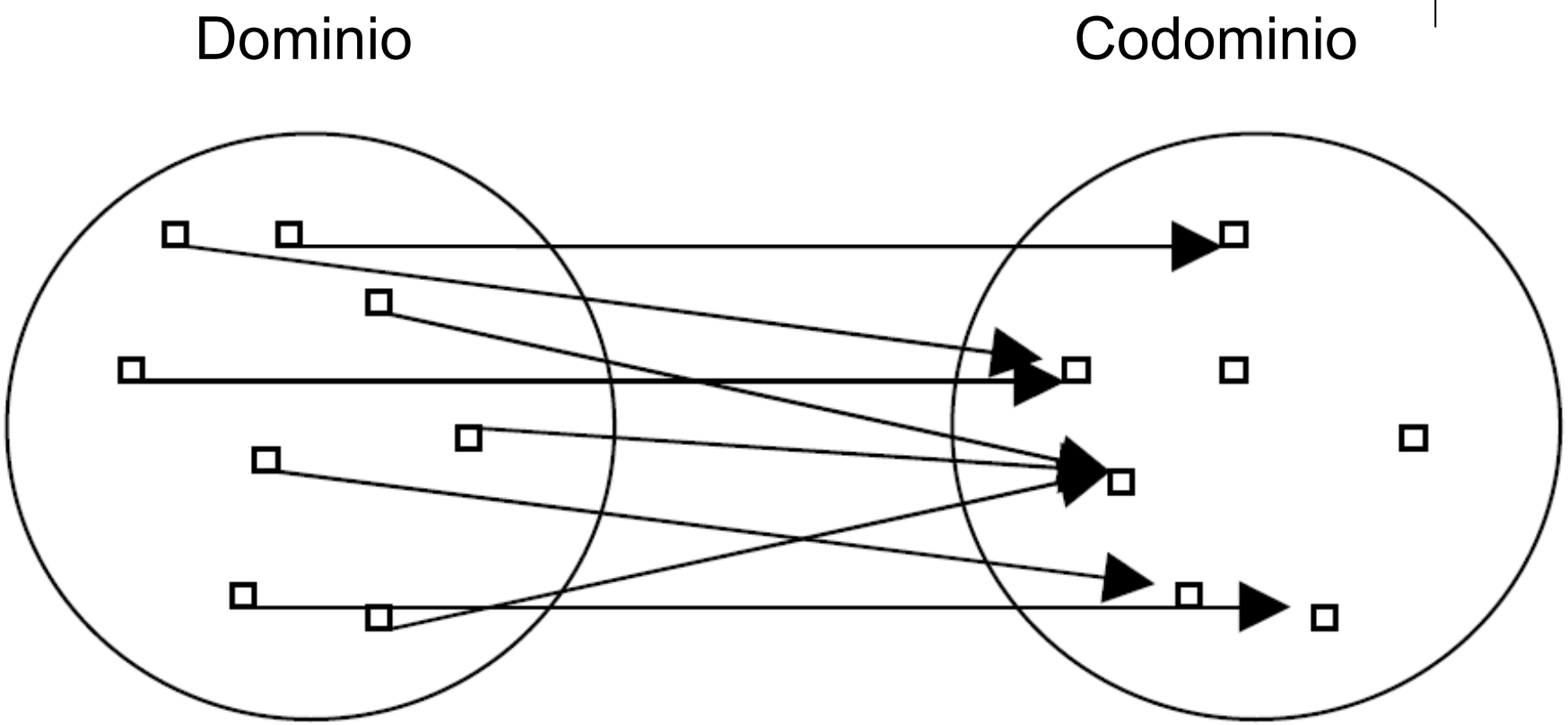
- ALGORITMO per calcolare Fact(n)

```
se n = 0 allora restituisci 1
  altrimenti restituisci n × fact(n − 1)
```

- Programma in CaML

```
let rec fact n =
        if n=0 then 1
        else n * fact(n-1)
```

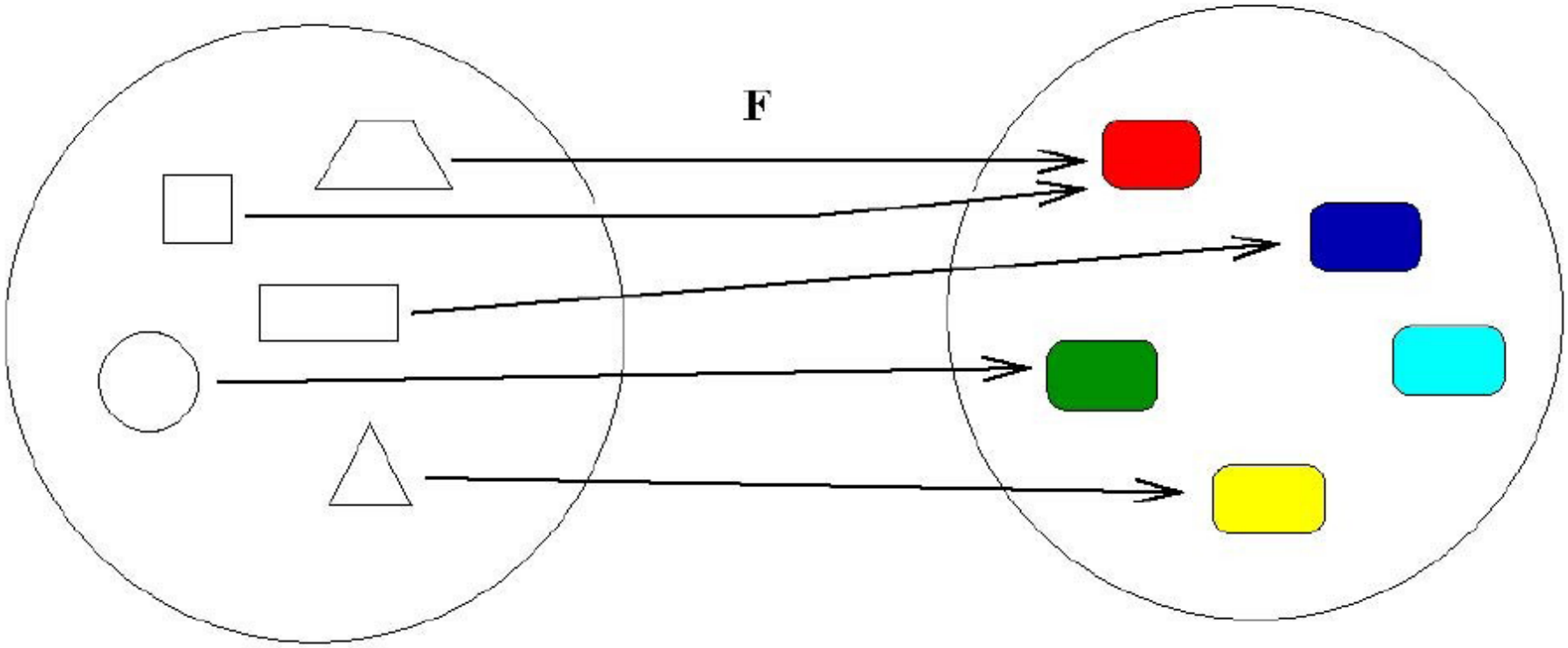
Funzioni



F associa a ogni elemento del DOMINIO uno ed un solo elemento del CODOMINIO

Il TIPO di F è: DOMINIO → CODOMINIO

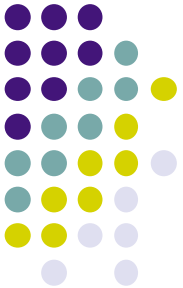
Esempio di funzione



F : FORME → COLORI

F si applica a un elemento di FORME (argomento della funzione) e RESTITUISCE un elemento di COLORI come VALORE

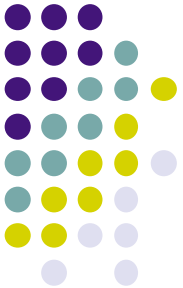
Tipi



- COS'E' UN TIPO?
  - Un tipo descrive un insieme di valori e le operazioni che si possono fare su di essi
- Se A è un tipo e $x \in A$, diciamo che x è di tipo A

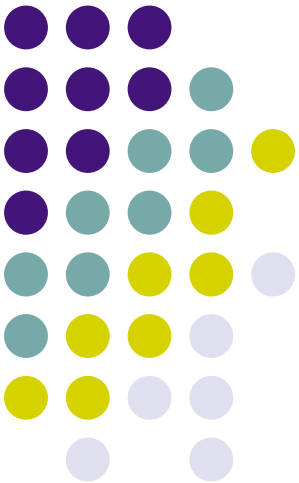
  $x : A$
- Ad esempio:
  - 3 : INT
  - (2, 5) : INT × INT
  - (1, 2, 3) : INT × INT × INT oppure (1, 2, 3) : INT^3
- Il tipo $A \rightarrow B$ è l'insieme di tutte le funzioni che hanno A come dominio e B come codominio.

Linguaggio funzionale ML

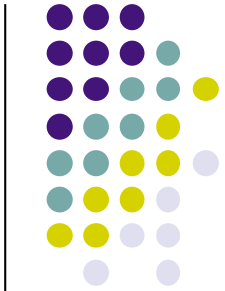


- ML nasce come linguaggio di specifica per strategie di prova. Poi ci si accorge che può essere utilizzato per programmare
- ML è un linguaggio fortemente tipato: ogni espressione in ML ha un tipo che può essere determinato a tempo di compilazione. Il compilatore può escludere alcuni errori dinamici.
- ML ha un meccanismo di inferenza di tipi che gli consente di dedurre qual è il tipo di un'espressione senza bisogno di dichiarazioni esplicite.
- ML ha un sistema di tipi polimorfo: una funzione può accettare argomenti di vari tipi.

Perchè la programmazione funzionale ?



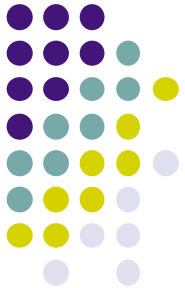
Programmazione alla portata di tutti



- In un’epoca di comunicazione di massa, di diffusione di massa dell’informatica e della cultura in generale, è quasi implicito che anche la programmazione sia passata da scienza a cultura di base e sia divenuta alla portata di tutti.

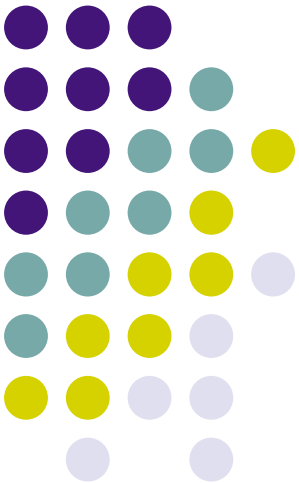
- Linguaggi di programmazione di ogni genere sono largamente diffusi ed usati; ciò nonostante, invece di avere un forte progresso della qualità dei programmi, abbiamo avuto dagli ultimi dieci anni uno spaventoso incremento degli errori di programmazione.

Correttezza dei programmi

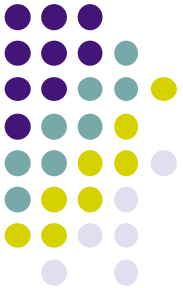


- L'unico metodo efficace per affermare che un programma è esente da errori è la dimostrazione di correttezza rispetto alla specifica, ma questa è indecidibile ed è molto complesso riuscire a dimostrare, anche a mano, la correttezza di un lungo programma.
- Si possono invece limitare, ma non evitare del tutto, gli errori di programmazione fornendo una dimostrazione di correttezza parziale, cioè relativa solo a parte della specifica.
- Una dimostrazione di correttezza parziale molto usata (anche inconsciamente) è il controllo di tipi statico: si specificano nel programma i tipi e il compilatore riesce a controllare che i tipi effettivi siano conformi a quelli dichiarati.

Programma del corso



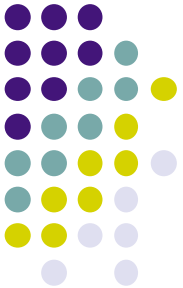
Parte 1



- Introduzione al problem solving.
- Introduzione alla programmazione funzionale e al linguaggio CaML.
- Tecniche di risoluzione dei problemi: approccio ricorsivo ed iterativo.
- Procedure come astrazione di istruzioni.
- Procedure ricorsive.
- Verfica della correttezza dei programmi ricorsivi attraverso dimostrazioni per induzione.
- Verifica della correttezza dei programmi iterativi attraverso il concetto di invariante.
- Liste.

Le implementazioni del linguaggio CaML adotatte sono: CaML/INRIA e F#/ Microsoft Research.

Parte 2



- Introduzione al linguaggio ANSI C.
- Tipi di dati predefiniti: int, double, char.
- Istruzioni: assegnamento, selezione, ripetizione.
- Procedure e Funzioni: dichiarazione e definizione, passaggio di parametri per valore e per indirizzo, funzioni ricorsive, funzioni di ordine superiore.
- Tipi di dati predefiniti, tipi enumerati, array, stringhe, strutture e puntatori.
- Memoria Dinamica: allocazione e deallocazione di memoria.
- Liste semplici, doppie, circolari.
- Input-Output: procedure di libreria sugli standard IO.
- I file: il tipo FILE, creazione e manipolazione di file.