

# La ricorsione con **Haskell**

Luca Barra

Anno accademico 2023/2024



# INDICE

<b>CAPITOLO 1</b>	<b>INTRODUZIONE</b>	<b>PAGINA 2</b>
1.1	Livello di scuola, classe e indirizzo	2
1.2	Motivazioni e finalità	2
1.3	Prerequisiti	3
1.4	Contenuti	3
1.5	Traguardi e obiettivi Obiettivi di apprendimento — 3 • Indicazioni nazionali — 4	3
1.6	Materiali e strumenti necessari	4
1.7	Linguaggio	4
<b>CAPITOLO 2</b>	<b>SVILUPPO DEI CONTENUTI</b>	<b>PAGINA 7</b>
2.1	Attività 1: Introduzione alla ricorsione	7
2.2	Attività unplugged Attività 2: Le matrioske — 8 • Attività 3: Da Unplugged a Programmazione — 8	8
2.3	Esercizi di programmazione Esempio — 9 • Esercizi per prendere confidenza — 9 • Esercizi sulle guardie — 9 • Esercizi sulle liste — 11	9
<b>CAPITOLO 3</b>	<b>GUIDA PER GLI INSEGNANTI</b>	<b>PAGINA 13</b>
3.1	Consigli sull'utilizzo del materiale didattico	13
3.2	Snodi e indicatori per fasi Attività 1 — 13 • Attività 2 — 13 • Attività 3 — 13	13
<b>CAPITOLO 4</b>	<b>INDICAZIONI PER LA VALUTAZIONE</b>	<b>PAGINA 15</b>
4.1	Valutazione durante le lezioni/attività	15
4.2	Riguardo gli esercizi di programmazione... Block Model — 15 • Misconceptions — 15	15
4.3	Rubrica valutativa	15



# Capitolo 1

## Introduzione

### Note:-

Quest'attività, essendo pesante e impegnativa, richiede 2-3 lezioni per essere svolta (volendo si può pensare a una quarta lezione per approfondire meglio).

### 1.1 Livello di scuola, classe e indirizzo

#### Domanda 1

A chi è rivolta questa attività?

**Risposta:** A studenti del quarto/quinto anno di una scuola secondaria di secondo grado di un indirizzo **scientifico**<sup>1</sup>.

#### Domanda 2

Può essere adattata/rivolta a studenti di diverse età e indirizzi?

**Risposta:** Quest'attività può essere somministrata a studenti al quinto anno di superiori senza alcuna modifica. Può, altresì, essere eseguita da studenti di età inferiore con alcune correzioni (che indicherò nel documento).

### 1.2 Motivazioni e finalità

#### Domanda 3

Perché si è scelta proprio quest'attività?

**Risposta:** L'attività nasce dalla necessità di insegnare ai ragazzi delle superiori il concetto di programma ricorsivo. Spesso quest'argomento viene trattato in modo superficiale o non viene trattato proprio, ma è utile avere un modello mentale di ricorsione per poter vedere i problemi sotto un'altra luce. Infatti lo scopo di questa unità didattica non è solamente quello di insegnare un argomento, ma punta a mostrare soluzioni alternative ad alcuni problemi.

⇒ **Scrivere codice in modo elegante:** quando si insegna a programmare, almeno all'inizio, viene trascurato il fatto che il codice debba essere letto e capito da altre persone per cui ci si focalizza più sull'aspetto "funziona" rispetto alla "veste grafica". Questo può anche portare a errori di programmazione dato che si avranno difficoltà a leggere anche i propri programmi. Il presente documento non si vuole semplicemente limitare a insegnare il concetto, molto utile, della ricorsione, ma vuole anche far fronte al problema dello "spaghetti code" (programmi mal strutturati) fornendo una valida alternativa.

---

<sup>1</sup>Liceo Scientifico Op. Scienze Applicate

⇒ **Pensiero computazionale e algoritmico:** una caratteristica fondamentale nella società odierna è quella di far fronte a problemi spesso imprevisti. Non c'è solo un modo per risolvere un quesito e le soluzioni possono essere molteplici. Infatti, per alcuni problemi, può risultare ostico usare un approccio iterativo per cui è preferibile utilizzare la ricorsione.

## 1.3 Prerequisiti

Verranno dati per scontati i seguenti prerequisiti in quanto fondamentali per la comprensione dell'unità didattica.

- ⇒ Utilizzo di base del computer;
- ⇒ Concetto di algoritmo;
- ⇒ Concetto di variabile;
- ⇒ Tipi di variabili (Int, Float, etc.);
- ⇒ Logica booleana;
- ⇒ Propensione al ragionamento astratto.

## 1.4 Contenuti

I contenuti che presentano un asterisco blu (\*) sono adatti anche a studenti di altri indirizzi o di età inferiore (quarta superiore). I contenuti che presentano un asterisco rosso (\*) sono opzionali per via della maggiore difficoltà, ma se gli alunni reagiscono bene al resto dell'attività si potrebbero considerare come "bonus".

- ⇒ \* Ricorsione;
- ⇒ \* Funzioni;
- ⇒ \* Tipi primitivi di Haskell;
- ⇒ Inferenza di tipo;
- ⇒ Funzioni a più argomenti;
- ⇒ Funzioni con guardie;
- ⇒ Liste;
- ⇒ Pattern matching;
- ⇒ \* Funzioni anonime;
- ⇒ \* Alberi.

## 1.5 Traguardi e obiettivi

### 1.5.1 Obiettivi di apprendimento

**Note:-**

I seguenti obiettivi di apprendimento sono stati scritti usando la tassonomia di Bloom rivisitata.

Tassonomia	Obiettivi
<b>CREATE</b>	Lo/La studente/ssa, al termine dell'attività, sarà in grado di <b>sviluppare</b> semplici programmi ricorsivi in Haskell.
<b>EVALUATE</b>	Lo/La studente/ssa, al termine dell'attività, saprà <b>valutare</b> la convenienza di un approccio ricorsivo rispetto a un approccio iterativo.
<b>ANALYZE</b>	Lo/La studente/ssa, al termine dell'attività, comprenderà i concetti di passo base e passo induttivo/ricorsivo e saprà <b>simulare</b> l'esecuzione di programmi che usano esplicitamente la ricorsione.
<b>APPLY</b>	Lo/La studente/ssa, al termine dell'attività, riuscirà a <b>usare</b> in modo efficace la ricorsione per risolvere problemi.
<b>UNDERSTAND</b>	Lo/La studente/ssa, al termine dell'attività, saprà <b>identificare, classificare e descrivere</b> programmi ricorsivi.
<b>REMEMBER</b>	Lo/La studentessa, al termine dell'attività, <b>ricorderà</b> le componenti fondamentali di un programma ricorsivo (passo base e passo induttivo).

### 1.5.2 Indicazioni nazionali

**Ambito algoritmico:** implementazione di un linguaggio di programmazione, metodologie di programmazione.

**Ambito calcolo numerico e simulazioni:** semplici simulazioni.

## 1.6 Materiali e strumenti necessari

**Materiale fisico:**

- ⇒ Matrioske (bambole russe);
- ⇒ Lavagna multimediale;
- ⇒ Computers.

**Materiale software:**

- ⇒ GHCi (interprete per Haskell);
- ⇒ Editor di testo o IDE (Integrated Development Enviroment);
- ⇒ Terminale o Powershell.

**Note:-**

GHCI può essere reperito al seguente link: <https://www.haskell.org/downloads/>. Si raccomanda di utilizzarlo su un computer con Linux, anche se rimane comunque installabile su Windows.

## 1.7 Linguaggio

**Linguaggio scelto:** *Haskell*.

### Domanda 4

Perché si è scelto questo linguaggio?

***Risposta:*** Haskell è un linguaggio funzionale puro, il che lo rende ideale per insegnare agli studenti un concetto importante come la ricorsione. Inoltre Haskell presenta una sintassi molto semplice e pulita, che non distrae gli studenti dal concetto che si sta insegnando: i codici risultano molto più compatti rispetto ad altri linguaggi (C, Java, ecc...). Infine, Haskell è un linguaggio unico nel suo genere, in quanto "lazy" e "strongly typed", il che lo rende un linguaggio molto interessante da studiare e da approfondire.





## Capitolo 2

# Sviluppo dei contenuti

### 2.1 Attività 1: Introduzione alla ricorsione

#### Per gli insegnanti 1

Bisogna inizialmente introdurre in linea generale il contenuto dell'attività agli studenti, facendo attenzione a non perdersi nei dettagli che verranno approfonditi durante le fasi successive. Questo aiuta ad accendere la curiosità degli studenti e a farli partecipare attivamente all'attività.

#### Note:-

In questo documento ho inserito note specifiche per gli insegnanti (come quella sopra). Tuttavia sono solo brevi suggerimenti, in quanto la guida per gli insegnanti vera e propria è contenuta nella sezione seguente.

#### Materiale utilizzato in quest'attività:

- ⇒ Lavagna multimediale;
- ⇒ Computers.

#### Per gli insegnanti 2

In questa fase i computers sono opzionali. Il docente può, a discrezione personale, decidere di condividere sui computers degli studenti il codice di esempio (per una maggiore interazione) oppure mostrarlo sulla lavagna multimediale.

#### Fase 1:

- ⇒ **Consegna:** come prima cosa, per introdurre quest'attività si consiglia di ripassare brevemente i tipi di dati (Int, float, etc.). Successivamente il docente introduce la ricorsione spiegando i concetti di passo base e passo ricorsivo. Per supportare quest'attività si può fare riferimento al primo esercizio di programmazione proposto (Fibonacci ricorsivo). Al termine di questa fase si dovranno anche introdurre gli elementi sintattici basilari di Haskell.
- ⇒ **Svolgimento:** inizialmente si presenterà la ricorsione in maniera discorsiva anche facendo esempi legati alla quotidianità (per esempio un **treno**: se ha 1 solo vagone, passo base, allora è semplicemente un vagone. Se ha  $n > 1$  vagoni allora è un treno composto da una "testa" e da  $n - 1$  vagoni). Eventualmente si possono anche mostrare, su una lavagna multimediale, immagini che presentano la ricorsione (per esempio alcuni dei quadri di Escher come "Mani che disegnano" o "Cascata"). Dopo di che gli studenti verranno incoraggiati a partecipare portando ulteriori esempi di ricorsività sulla base delle loro esperienze personali. Infine si mostra l'esempio riguardante Fibonacci, scritto in Haskell e si approfittà per dare un'idea concreta di utilizzo di ricorsione.
- ⇒ **Discussione:** il docente dovrà gestire una discussione sulla base degli esempi portati dagli studenti e correggere eventuali errori concettuali tramite il dialogo.

⇒ **Conclusione:** Per concludere quest'attività il docente introdurrà la sintassi base di Haskell (insieme all'utilizzo di GHCi) e si assicurerà che essa venga compresa dagli studenti.

### Per gli insegnanti 3

Può essere utile, prima di proseguire con l'attività unplugged, porre alcune domande agli studenti per assicurarsi che abbiano compreso le basi su cui si sviluppa la ricorsione. Si può prendere spunto dalla sezione riguardante gli "Indicatori" presente nel capitolo successivo.

#### Note:-

Comandi utili di GHCi:

- `:load` o `:l`, serve per compilare un file `hs` (Haskell);
- `:type` o `:t`, mostra il tipo di una funzione;
- `:reload` o `:r`, ricompila tutti i file precedentemente compilati;
- `:!clear`, pulisce lo schermo del terminale di GHCi (solo linux).

## 2.2 Attività unplugged

### 2.2.1 Attività 2: Le matrioske

### 2.2.2 Attività 3: Da Unplugged a Programmazione

## 2.3 Esercizi di programmazione

### 2.3.1 Esempio

**Fibonacci ricorsivo:** viene usato nella prima parte dell'attività come esempio di utilizzo della ricorsione.

```
1 fibonacci :: Int -> Int
2 fibonacci 0 = 0
3 fibonacci 1 = 1
4 fibonacci n = fibonacci (n - 1) + fibonacci (n - 2)
```

### 2.3.2 Esercizi per prendere confidenza

**Somma:** Scrivere una funzione in Haskell (con tipo `Int -> Int`), che prenda in input un intero `n` e calcoli la somma dei primi `n` numeri naturali.

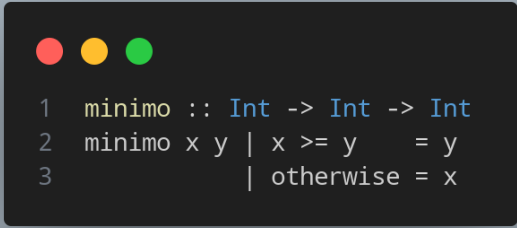
```
1 -- Un'applicazione si può scrivere in notazione infissa usando i backticks.
2 -- Per esempio la somma dei primi n numeri naturali si può scrivere come:
3
4 somma :: Int -> Int
5 somma n = n * (n + 1) `div` 2
6
7 -- Un'applicazione si può scrivere in notazione prefissa.
8 -- Per esempio la somma dei primi n numeri naturali si può scrivere come:
9
10 somma2 :: Int -> Int
11 somma2 n = div (n * (n + 1)) 2
```

### 2.3.3 Esercizi sulle guardie

**Massimo:** Scrivere una funzione in Haskell (con tipo `Int -> Int -> Int`), usando le guardie, che prenda in input due interi e restituisca il maggiore.

```
1 massimo :: Int -> Int -> Int
2 massimo x y | x >= y    = x
3              | otherwise = y
```

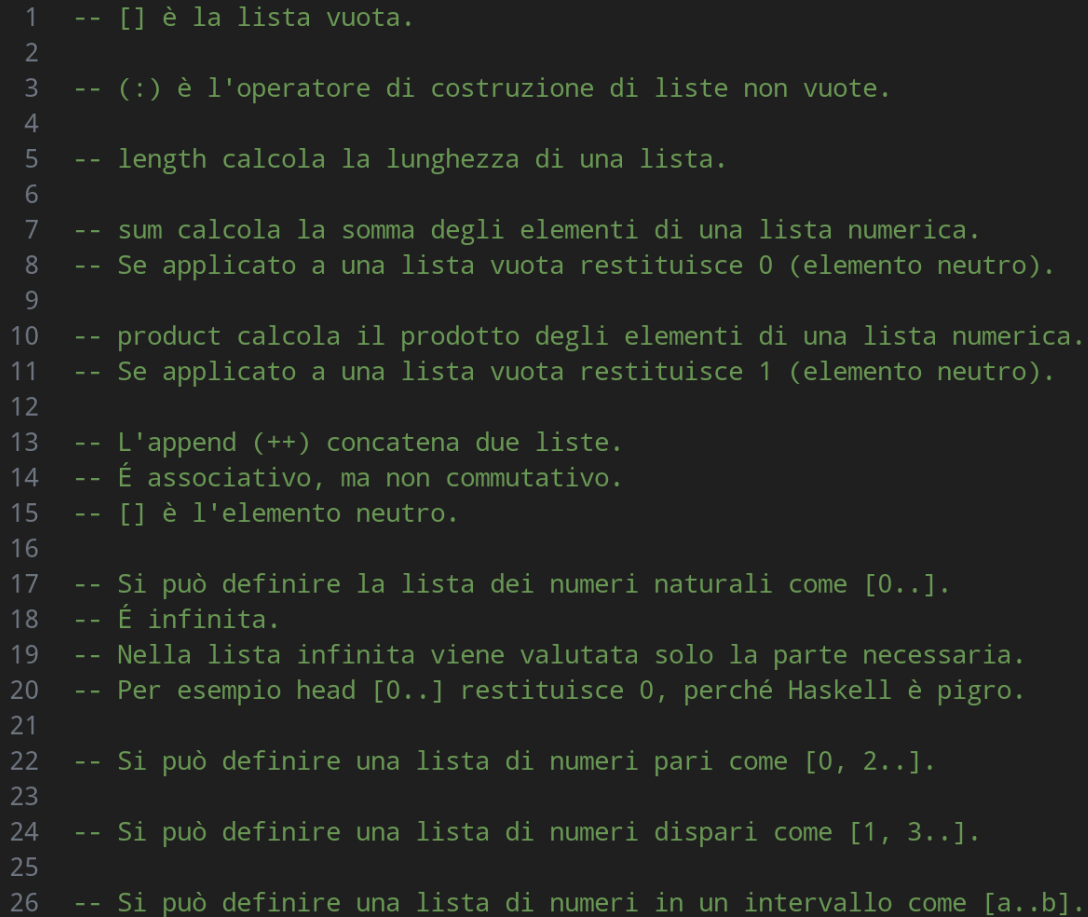
**Massimo:** Scrivere una funzione in Haskell (con tipo `Int -> Int -> Int`), usando le guardie, che prenda in input due interi e restituisca il minore.



```
1 minimo :: Int -> Int -> Int
2 minimo x y | x >= y    = y
3             | otherwise = x
```

### 2.3.4 Esercizi sulle liste

Prima di iniziare gli esercizi con le liste si deve condividere questa "dispensa" che illustra le principali caratteristiche delle liste.



```
1  -- [] è la lista vuota.
2
3  -- (:) è l'operatore di costruzione di liste non vuote.
4
5  -- length calcola la lunghezza di una lista.
6
7  -- sum calcola la somma degli elementi di una lista numerica.
8  -- Se applicato a una lista vuota restituisce 0 (elemento neutro).
9
10 -- product calcola il prodotto degli elementi di una lista numerica.
11 -- Se applicato a una lista vuota restituisce 1 (elemento neutro).
12
13 -- L'append (++) concatena due liste.
14 -- É associativo, ma non commutativo.
15 -- [] è l'elemento neutro.
16
17 -- Si può definire la lista dei numeri naturali come [0..].
18 -- É infinita.
19 -- Nella lista infinita viene valutata solo la parte necessaria.
20 -- Per esempio head [0..] restituisce 0, perché Haskell è pigro.
21
22 -- Si può definire una lista di numeri pari come [0, 2..].
23
24 -- Si può definire una lista di numeri dispari come [1, 3..].
25
26 -- Si può definire una lista di numeri in un intervallo come [a..b].
```



## Capitolo 3

# Guida per gli insegnanti

### 3.1 Consigli sull'utilizzo del materiale didattico

### 3.2 Snodi e indicatori per fasi

#### 3.2.1 Attività 1

##### *Fase 1:*

- Snodi:
  - ⇒ Capire la struttura di una funzione ricorsiva;
  - ⇒ Analizzare un programma ricorsivo;
  - ⇒ Comprensione della sintassi di Haskell.
- Indicatori:
  - ⇒ Quali sono le caratteristiche di una funzione ricorsiva?
  - ⇒ Identificare il passo base e il passo ricorsivo nel programma "Fibonacci ricorsivo".
  - ⇒ Elencare i principali tipi di Haskell.

#### 3.2.2 Attività 2

##### *Fase 1:*

- Snodi:
  - ⇒
- Indicatori:
  - ⇒

#### 3.2.3 Attività 3

##### *Fase 1:*

- Snodi:
  - ⇒
- Indicatori:
  - ⇒





## Capitolo 4

# Indicazioni per la valutazione

### 4.1 Valutazione durante le lezioni/attività

### 4.2 Riguardo gli esercizi di programmazione...

#### 4.2.1 Block Model

#### 4.2.2 Misconceptions

### 4.3 Rubrica valutativa

Questa piccola rubrica valutativa può essere utilizzata nell'ambito della fase di esercitazione in Haskell come linea guida di valutazione.

	Assente	Parziale	Adeguate
<b>Comprensione teorica</b>	Conoscenza della teoria gravemente insufficiente e/o assente.	Conoscenza della teoria sufficiente.	Conoscenza della teoria completa e/o approfondita.
<b>Sintassi di Haskell</b>	Mancanza di comprensione della sintassi di Haskell.	Comprensione della sintassi base di Haskell (variabili, guardie, etc.), difficoltà con concetti come "Inferenza di tipo" e/o "Pattern matching".	Ottima padronanza di Haskell e di GHCi.
<b>Esercizi</b>	Esercizi non svolti e/o svolti in maniera scorretta.	Esercizi svolti in maniera parzialmente corretta, ma con qualche lacuna.	Esercizi svolti correttamente rispetto alle consegne.
<b>Padronanza della terminologia tecnica</b>	Terminologia assente e/o totalmente inadeguata.	Terminologia sostanzialmente corretta, ma con alcune imprecisioni.	Terminologia pienamente corretta.
<b>Comprensione della ricorsione</b>	Assenza del concetto di ricorsione e/o idea completamente sbagliata della ricorsività.	Comprensione basilare della ricorsione.	Comprensione eccellente della ricorsione e delle sue implicazioni.

