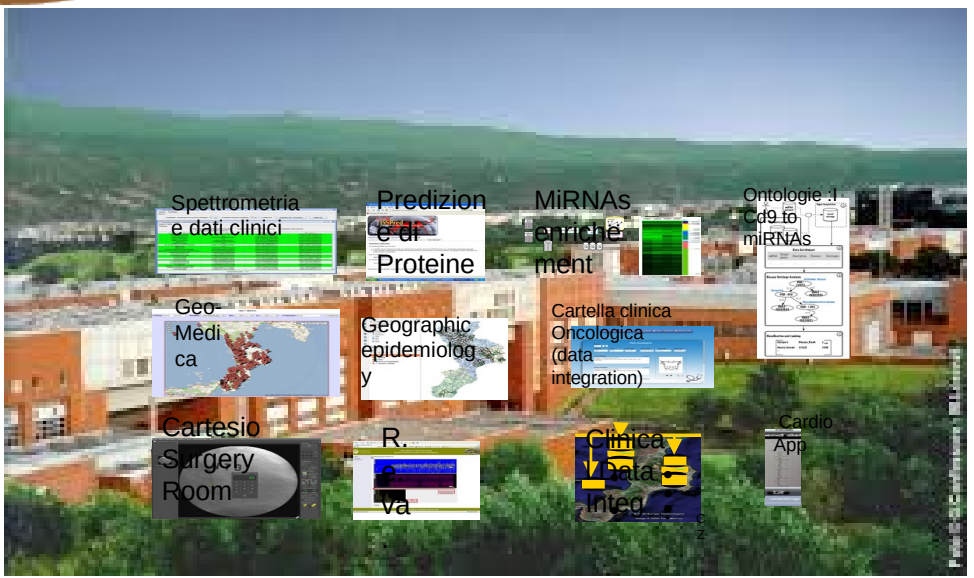
Corso di Tecniche di Programmazione
cdl Medicina e Chirurgia TD
II anno



Informazioni utili

- Corso del II anno II semestre
- Docente: Pierangelo Veltri (pierangelo.veltri@dimes.unical.it)
  - Ing Inf 06 – Bioingegneria Elettronica ed Informatica
- 3 CFU Lezione Frontale, 3 CFU tirocinio
- Corso di Ing-Inf/05 (Sistemi di Elaborazione delle Informazioni)
- Inquadramento dei corsi (rispetto a quanto fatto):
  - Informatica e Elementi di Informatica Medica (algoritmi, strutture dati elementari, vettori e matrici, algoritmi di allineamento, python…)
    - Informatica medica: tassonomie cliniche, terminologie per la medicina, cartelle cliniche
  - Basi di dati e sistemi informativi medici
  - Bioinformatica

Inquadramento e programma del corso di Tecniche di Programmazione

- Apprendimento di tecniche per lo sviluppo di algoritmi e soluzioni di problemi (es. *ricorsione*, *programmazione greedy*, *dinamica*, *divide et impera*, e *backtracking*) e della loro contestualizzazione nella progettazione di algoritmi risolutivi per problemi complessi.
- I contenuti sono presentati attraverso l'uso del linguaggio di programmazione Python, già introdotto nel corso di *Fondamenti di Informatica*. Al termine del corso gli studenti saranno in grado di risolvere problemi classici di ricerca e ottimizzazione, anche su grafi e di verificarne la soluzione attraverso l'uso del linguaggio Python.
- Fin qui..target...cerchiamo di comprendere la parte tecnologica e la parte medica

Informazioni utili

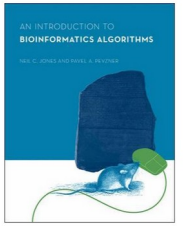
- Ricevimento (anche on line) compatibilmente all’orario.
- Proposta: Venerdi mattina ore 9:00 (in presenza cubo 44 Z) oppure on line (meet.google.com/smt-qnjb-ruu)
- Email: pierangelo.veltri@unical.it

Orario delle lezioni

- Lezione frontale: Venerdì dalle ore 14:30 alle ore 17:30 (3h)
  - aula 32B3
- Tirocinio: possibilità di incontri il lunedì dalle 16:30 alle 19:30 (3h), compatibilmente con il vostro orario
  - online/incontri a piccoli gruppi
  - Definizione di esercizi ed attività da definire (su argomenti di interesse medico/clinico)

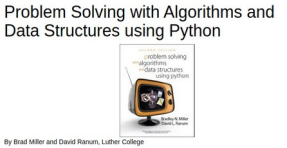
- **Link a Teams:**
- **Codice Teams: 6rsyrsf**

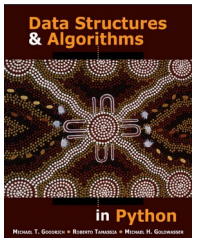
Materiale didattico



An Introduction to Bioinformatics Algorithms

di Neil C. Jones (Autore) Pavel A. Pevzner (Autore)





Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser:

Data Structures and Algorithms in Python. Computer Science (free book)

(**.pdf**)

Problem Solving with Algorithms and Data Structures using Python

Brad Miller and David Ranum, Luther College: (free, interactive book LINK

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduzione agli algoritmi e strutture dati***,** McGraw-Hil (pdf)

Per richiami su Python: How to Think Like a Computer Scientist: Interactive Edition, *https://runestone.academy/runestone/books/published/thinkcspy/index.html*

Algoritmi e strutture dati (Demetrescu et al.)

Libri di test

- Leggere le attività degli algoritmi in funzione dei risultati e delle applicazioni medico cliniche
- Es. introduzione alla bioinformatica ed all'health informatics
  - Definire meccanismi di gestione ottimale per dati relativi allo studio di pazienti e processi cardiologici
  - Definizione di meccanismi di gestione per l'analisi di campioni di urina e di spettrometria di massa (workflow management) in un processo di controllo e di analisi
  -

Programma (tentativo) e modalità d'esame

- Lezioni: 39 ore di didattica frontale + 30 ore di tirocinio/esercitazioni
- Lezione frontale: argomenti su strutture dati, algoritmi e tecniche di programmazione avanzate, mediante esempi, implementazioni in Python (ma non solo: introduzione anche ad altri ambienti di programmazione)
- Tirocinio: Esempi di applicazioni di tecniche di programmazione per la gestione di casi d'uso in ambito di interesse medico clinico
  - Ovvero: tecniche di programmazione come modalità di risoluzione di problemi di gestione dati ed informazioni di interesse biomedicale

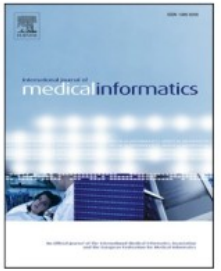
Medicina, Ingegneria Biotec

International Journal of Medical Informatics 172 (2023) 105002

Contents lists available at ScienceDirect

International Journal of Medical Informatics

journal homepage: www.elsevier.com/locate/ijmedinf



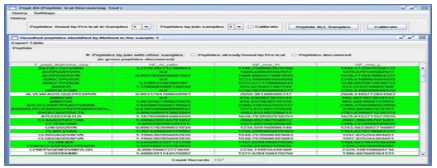
Bioengineering and medical informatics education in MD programs: perspectives from three Italian experiences

Riccardo Bellazzi [a], Maurizio Cecconi [b,c], Maria Laura Costantino [d], Pierangelo Veltri [e,f,*]

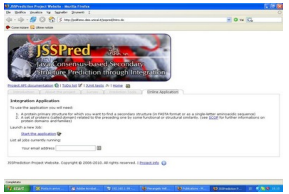
[a] *Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Italy*
[b] *Department of Biomedical Sciences, Anesthesiology and Intensive Care, Humanitas University Pieve Emanuele, Milan, Italy*
[c] *Anaesthesia and Intensive Care Medicine IRCCS Humanitas Research Hospital, Rozzano, Milan, Italy,*
[d] *Department of Chemistry, Materials and Chemical Engineering "Giulio Natta", Politecnico di Milano, Milan, Italy*
[e] *Department of Surgical and Medical Science, University Magna Graecia of Catanzaro, Catanzaro, Italy*
[f] *Computer Science, Modeling, Electronics, and Systems Engineering (DIMES), University of Calabria, Rende, Italy*

Qualche esempio …

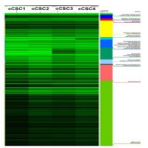
Spettrometria e dati clinici



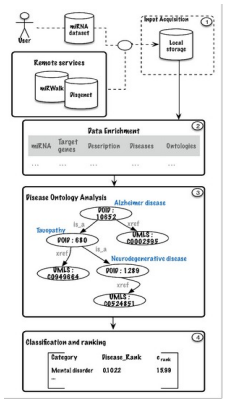
Predizione di Proteine



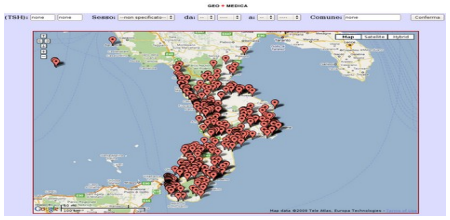
MiRNAs enrichement



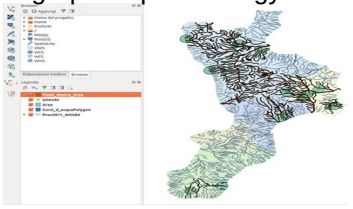
Ontologie :ICd9 to miRNAs



Geo-Medica



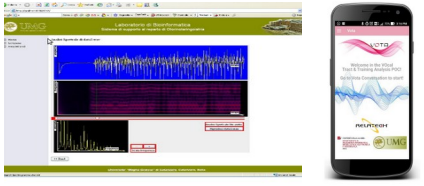
Geographic epidemiology



Cartella clinica Oncologica
(data integration)

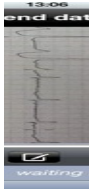
Cartesio Surgery Room

R.e.va.



Clinical Data Integ

Cardio App



Modalità d’esame (tentativo)

- Prova scritta: 2 esercizi su utilizzo di tecniche di programmazione per casi d’uso reale in ambito biomedicale (tentativo di test online con modulo google, soluzione in Python) – tempo a disposizione
- Ammissione a sostenere la prova orale con punteggio minimo di 18
- Ammissione con riserva con punteggio maggiore o uguale 15
- Prova orale: verifica sulla prova scritta, (eventuale), argomenti teorici trattati
- Il voto sarà determinato dai giudizi della prova scritta e della prova orale

Indice di programma (bozza)

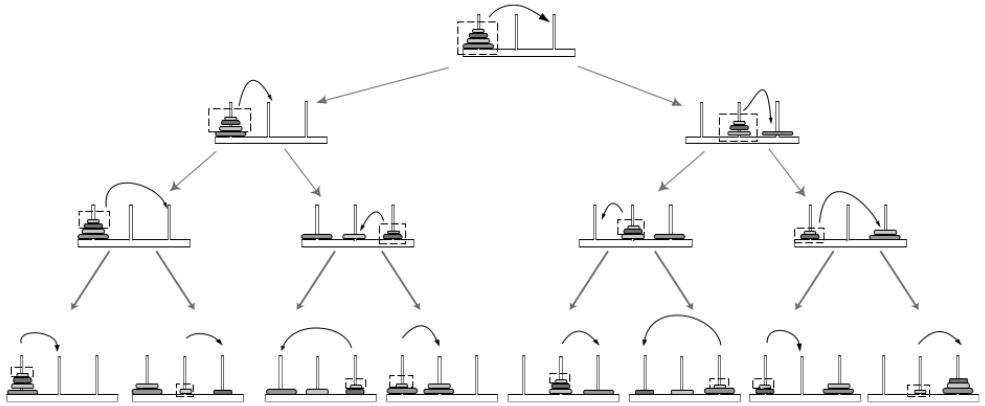
- Introduzione al corso e concetti preliminari, richiami di programmazione di base, il concetto di variabile, e la gestione della memoria (indirizzamento), compilatori di linguaggio ed esecuzione, modello a run time, struttura di un programma Python, complessità di calcolo
- Ricorsione, algoritmi di ricorsione e procedure di ordinamento, tecniche di ricorsione (es. Torre di Hanoi, …)
- Strutture dati dinamiche: array e liste, coda e pila, alberi e grafi
- Tecniche di programmazione evoluta: Divide et impera, tecnica golosa, programmazione dinamica, backtracking

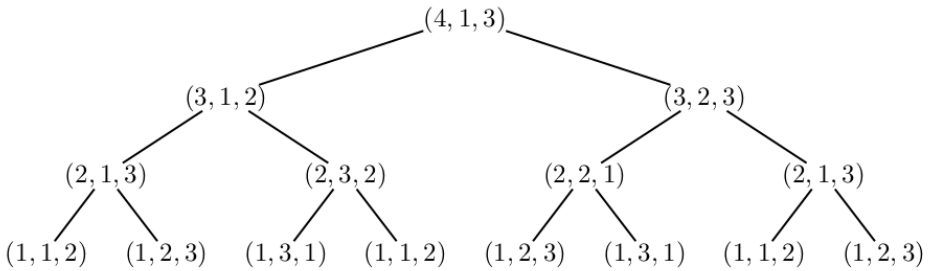
Esempio di ricorsione: La torre di Hanoi

Lo scopo del gioco è portare tutti i dischi su un paletto diverso, potendo spostare solo un disco alla volta e potendo mettere un disco solo su un altro disco più grande,
mai su uno più piccolo.

Link da wikipedia:

https://www.mathsisfun.com/games/towerofhanoi.html





Il problema si risolve in modo semplice
In termini di linee di codice in modo
Ricorsivo (vedremo)

"To solve a five-disk tower requires 31 moves, but to solve a hundred-disk tower would require more moves than there are atoms in the universe."

```
HANOITOWERS(n, fromPeg, toPeg)
1  if n = 1
2        output "Move disk from peg fromPeg to peg toPeg"
3        return
4  unusedPeg ← 6 − fromPeg − toPeg
5  HANOITOWERS(n − 1, fromPeg, unusedPeg)
6  output "Move disk from peg fromPeg to peg toPeg"
7  HANOITOWERS(n − 1, unusedPeg, toPeg)
8  return
```

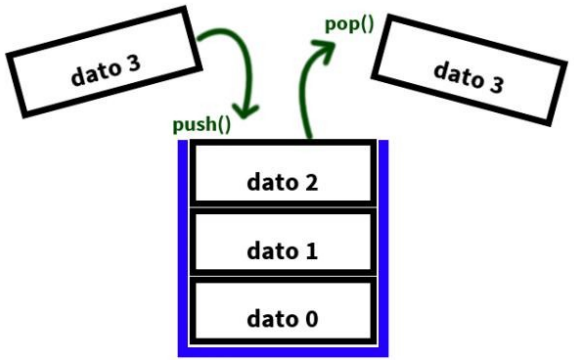
Lezione di oggi

- Richiami di programmazione
  - Variabili e Record di Attivazione
  - Strutture dati: vettori e matrici (in Python)
  - Istruzioni di controllo e cicli
  - ...

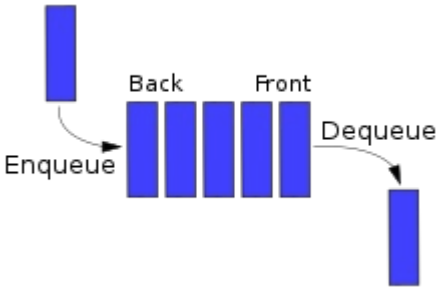
Strutture dati e tipi di dati (richiami)

- Esistono tipi di dati, dove il tipo indica la semantica interpretativa dei dati booleani memorizzati in memoria
  - Booleano, carattere, numeri
  - I linguaggi sono formali nell'indicazione del tipo (rigoroso e un po' piu complicato) oppure meno formali con identificazione del tipo a run time (python)
- Stutture dati:
  - Vettori, Record, Matrici, Alberi, Grafi, Pila (stack)
  - Ogni struttura dati serve a mappare informazioni da un livello logico ad un livello fisico con informazioni omogenee

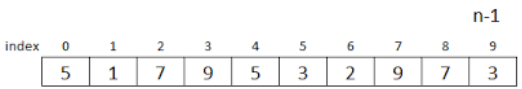
Esempi di strutture dati: rappresentazione logica e fisica



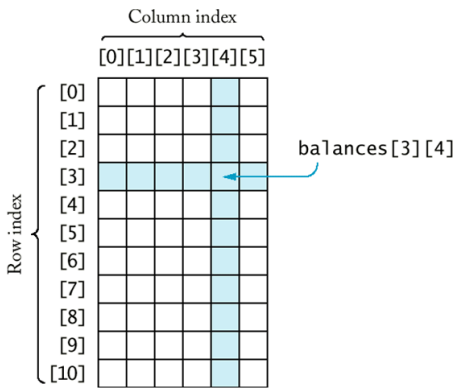
Pila (stack): inserimento e cancellazione dallo stesso punto



Coda: inserimento da un lato e rimozione dall'altro



Vettori e Matrici



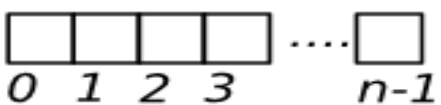
Rappresentazione fisica dei dati

- In memoria si accede sempre con un punto di partenza e un offset
- La gestione della memoria e' lasciata al S.O.
- L'allocazione e deallocazione è lasciata sempre al S.O.
- Alcuni linguaggi di programmazione consentono la gestione della memoria via programma
  - Esempio di caso d'uso: acquisizione di bioimmagini e gestione in memoria: il caso del simulatore (cartesio)

Esempi di strutture dati

I vettori

- Un *array* o *vettore* è una struttura dati omogenea, che contiene un numero finito n di elementi dello stesso tipo
- Si può immaginare un array come una sorta di casellario, le cui caselle sono dette *celle*
- Ciascuna delle celle si comporta come una variabile tradizionale; tutte le celle sono variabili di uno stesso tipo, detto *tipo base* dell'array
- Questi elementi sono individuati attraverso un *indice* numerico, che tipicamente va da 0 a $n - 1$



Esempi di vettori

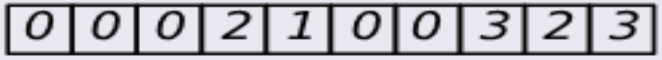
- Ad esempio, una stringa capace di contenere il nome di una persona (lunga fino ad un massimo di 15 caratteri):

Nome di persona

char nome[15]

- Il numero di tessera degli "Amici dell'UNICEF" è un vettore di 15 numeri interi

Tessera "Amici dell'UNICEF"



int numero tessera[10]

Accesso ad un elemento di un vettore

- Per accedere ad un elemento di un vettore (cioè leggerne il valore o assegnarvi un nuovo valore), utilizziamo l'operatore "[ ]"

Nome di persona

char a ← nome[0]	*Mette nella variabile a il carattere 'P'*
nome[1] ← 'i'	*Il nome diventa 'PiERPAOLO'*
nome[4] ← 4.2	*ERRORE! Il vettore può contenere solo caratteri*

I record

- Il *record* è una struttura dati che può essere eterogenea o omogenea, e quindi può contenere una combinazione di elementi che possono essere di diverso tipo, ad esempio un intero, un numero in virgola mobile e un carattere testuale
- Gli elementi di un record sono detti campi, e sono identificati da un nome
- Ad esempio

Numero complesso

```
struct numero complesso begin
    float parte reale
    float parte immaginaria
end struct
```

I record nella medicina

- Un ulteriore esempio di record – in un contesto medico – è la cartella clinica

Cartella clinica

```
struct cartella clinica begin
    int numero cartella
    char  paziente[100]
    ...
    struct diagnosi di ingresso begin
        char  data e ora[20]
        char  icd9[5]
        char  note[1000]
    end struct
    ...
end struct
```

Accesso ai contenuti di un record

- L'accesso ai campi di un record avviene usando l'operatore "."

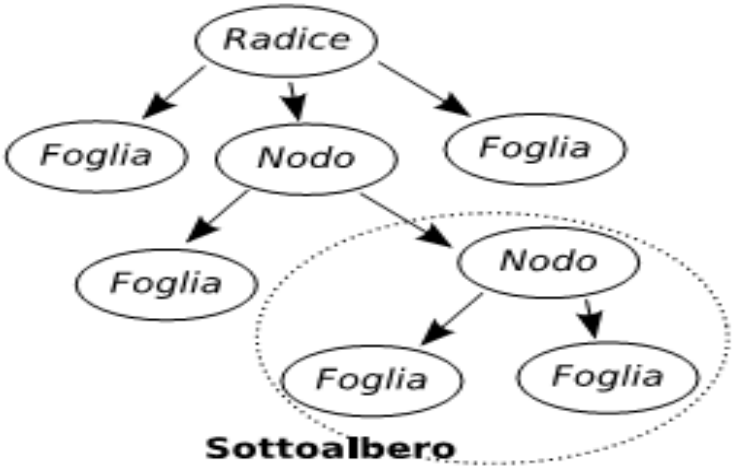
Accesso ai campi di un record

cartella clinica cc

int n ← cc.numero cartella
cc.paziente ← 'Pierpaolo Vittorini'
cc.diagnosi di ingresso.icd9 ← '540.9'

Gli alberi

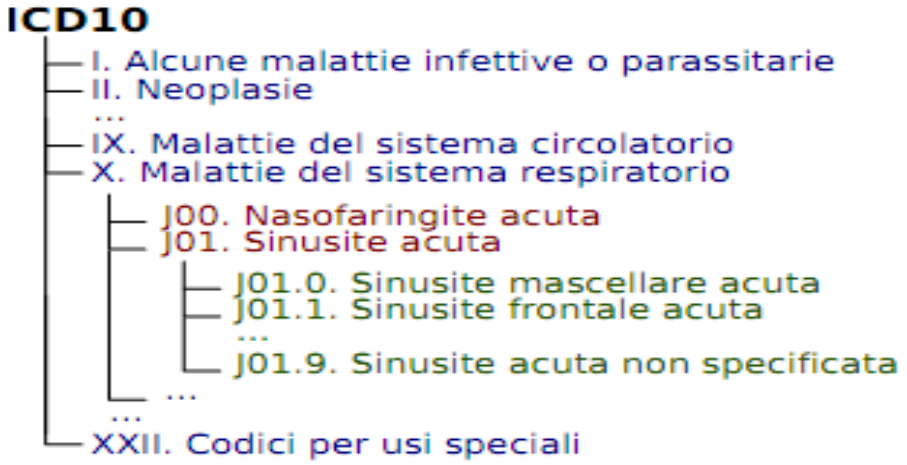
- Un albero è una struttura dati di tipo *gerarchico*
- In un albero, ciascun nodo ha un *padre* e un certo numero di *figli*
- Esiste però un nodo speciale chiamato *radice*, il quale non ha alcun padre



- Esistono altri nodi chiamati *foglie*, i quali non hanno figli
- Una porzione di albero è a sua volta un albero, chiamato *sottoalbero*
- Ciascun nodo porta con sè un'informazione

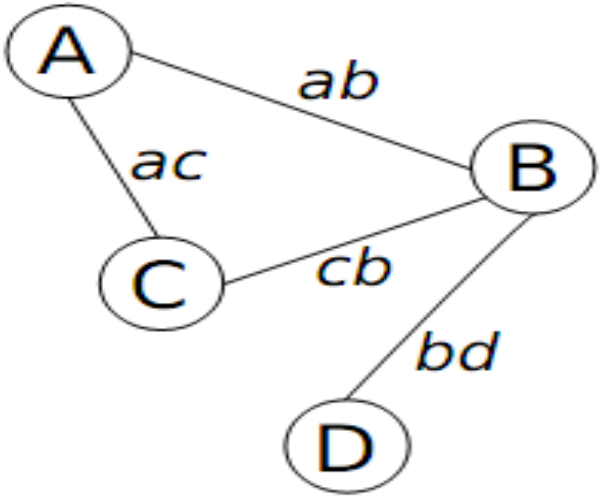
Gli alberi nella medicina

- Un esempio di uso di strutture ad albero nella medicina è relativa alla classificazione delle malattie



Grafi

- Un *grafo* è un insieme di elementi detti *nodi* o *vertici*, collegati fra loro da *archi* o *lati*, eventualmente dotati di *etichette*
- Il grafo è la struttura dati più complessa e ricca su cui l'informatica può far riferimento



Modello a run time per la gestione della memoria

- Il modello run time serve a simulare il comportamento della memoria durante l’esecuzione dei programmi
-

Il **modello runtime** è il modello al tempo di esecuzione dei programmi

- gestione della memoria e della capacità di elaborazione del calcolatore
- allocazione delle aree di memoria
- esecuzione delle operazioni da parte degli oggetti in caso programmazione a oggetti o dell'istanza in corso
- la gestione della memoria avviene per **aree di memoria**
- la gestione della memoria è dinamica
  - basata sulle operazioni di **allocazione** e **deallocazione** di aree di memoria

Esecuzione dei metodi

Il modello di gestione della memoria per l'esecuzione dei metodi (o costruttori) è basato sui **record di attivazione**

- un record di attivazione memorizza le informazioni necessarie a una singola attivazione di un metodo
  - punto di ritorno
  - riferimento all'oggetto esecutore
  - variabili locali
- esecuzione di un metodo
  - all'invocazione, viene allocato un record di attivazione
  - alla terminazione, il record di attivazione viene deallocato

I record di attivazione vengono gestiti mediante una **pila di attivazione**

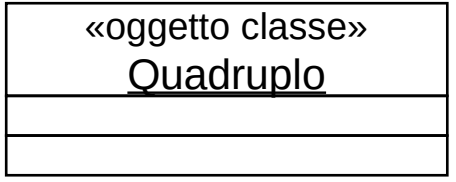
Esecuzione di Metodi

L'applicazione **Quadruplo**

```
class Quadruplo {
    public static int somma(int a, int b) {
        int c;                          // somma, 1
        c = a+b;                          // somma, 2
        return c;                         // somma, 3
    }
    public static int doppio(int n) {
        int d;                          // doppio, 1
        d = somma(n,n);                       // doppio, 2
        return d;                         // doppio, 3
    }
    public static void main(String[] args) {
        int a, b, c;                      // main, 1
        a = 2;                            // main, 2
        b = somma(a,a);   // b = 4              // main, 3
        c = doppio(b);    // c = 8              // main, 4
    }
}
```

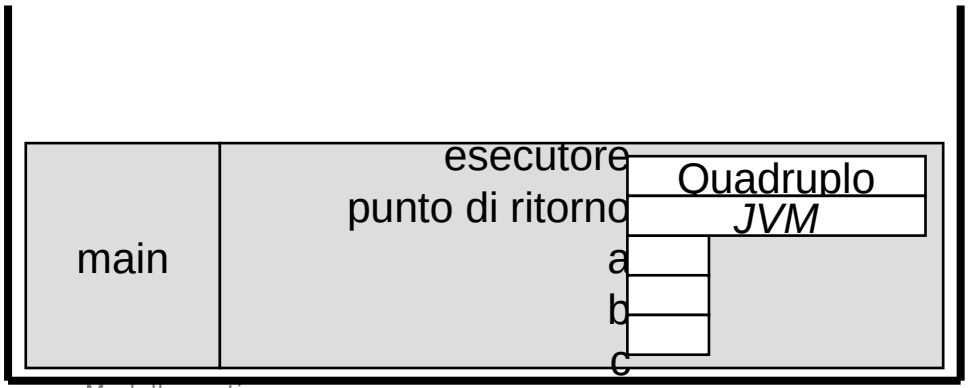
t=0 — avvio dell’esecuzione Quadruplo

L’esecuzione dell’applicazione **Quadruplo** viene richiesta mediante l’esecuzione del comando **java Quadruplo**



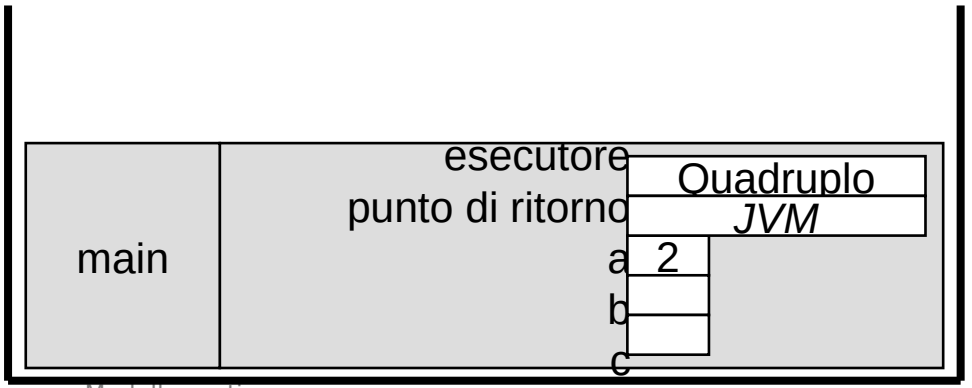
t=1 — attivazione di main

La JVM chiede all'oggetto **Quadruplo** di eseguire il metodo **main**



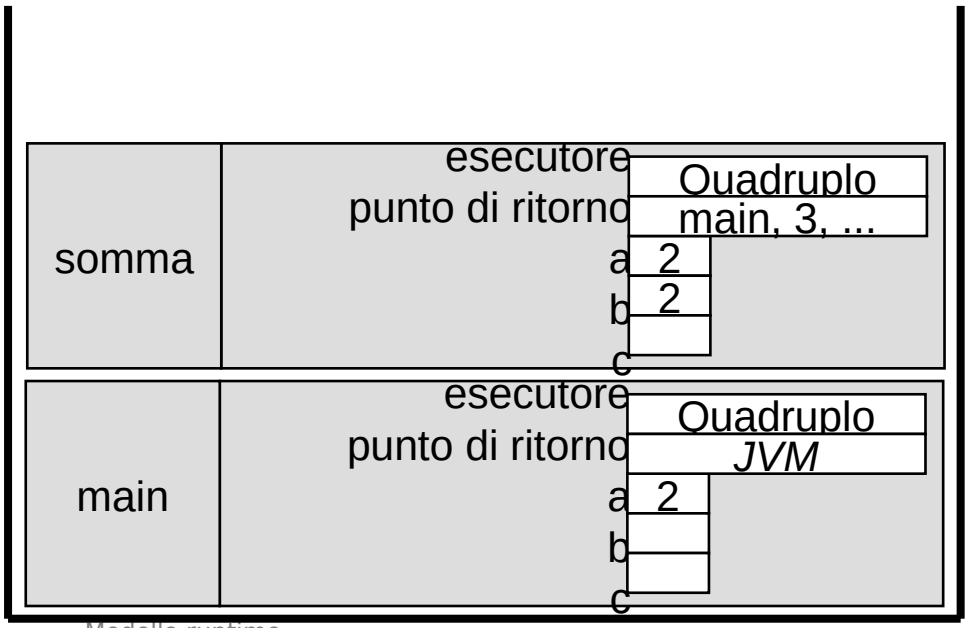
t=2 — assegnazione a una variabile

Viene eseguita l’assegnazione **a=2**



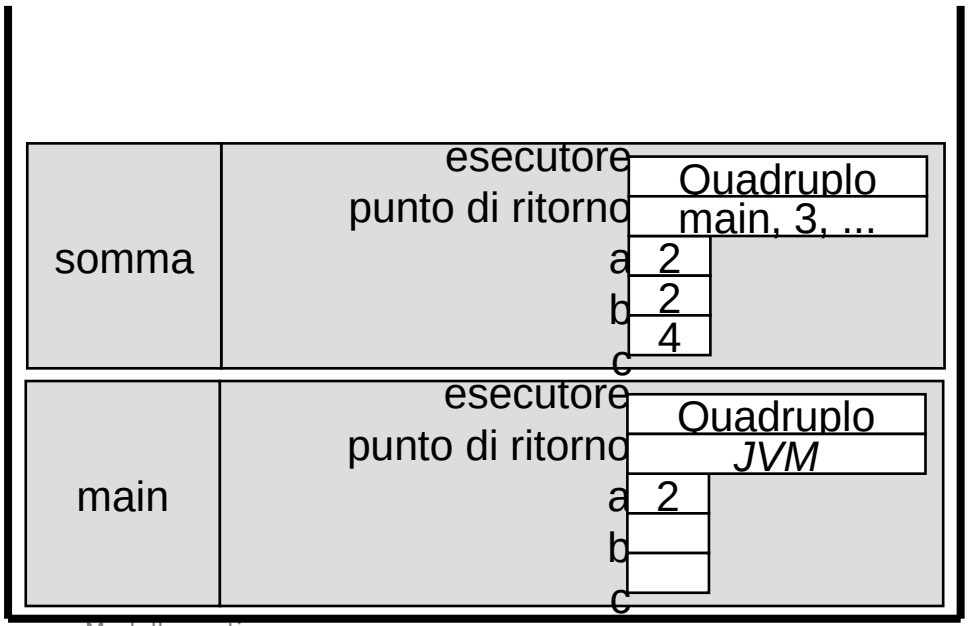
t=3 — invocazione e attivazione di somma

Il metodo **main** invoca il metodo **somma**



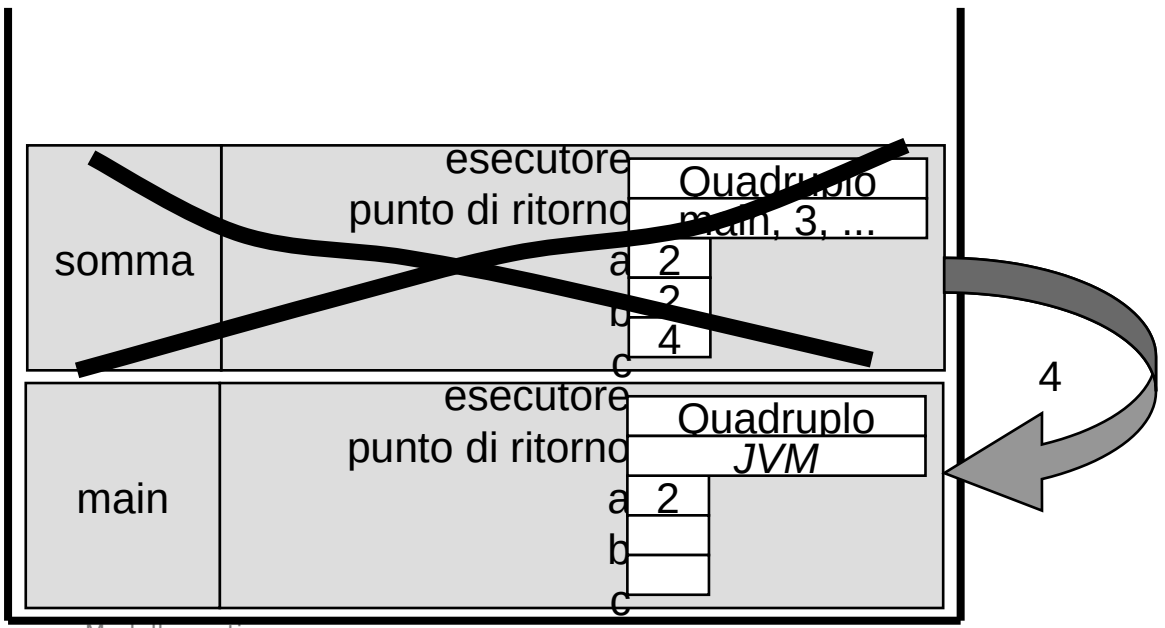
t=4 — esecuzione di una assegnazione

Viene eseguita l’assegnazione **c=a+b**



t=5 — terminazione di somma

Viene eseguita l'istruzione **return c** di **somma**

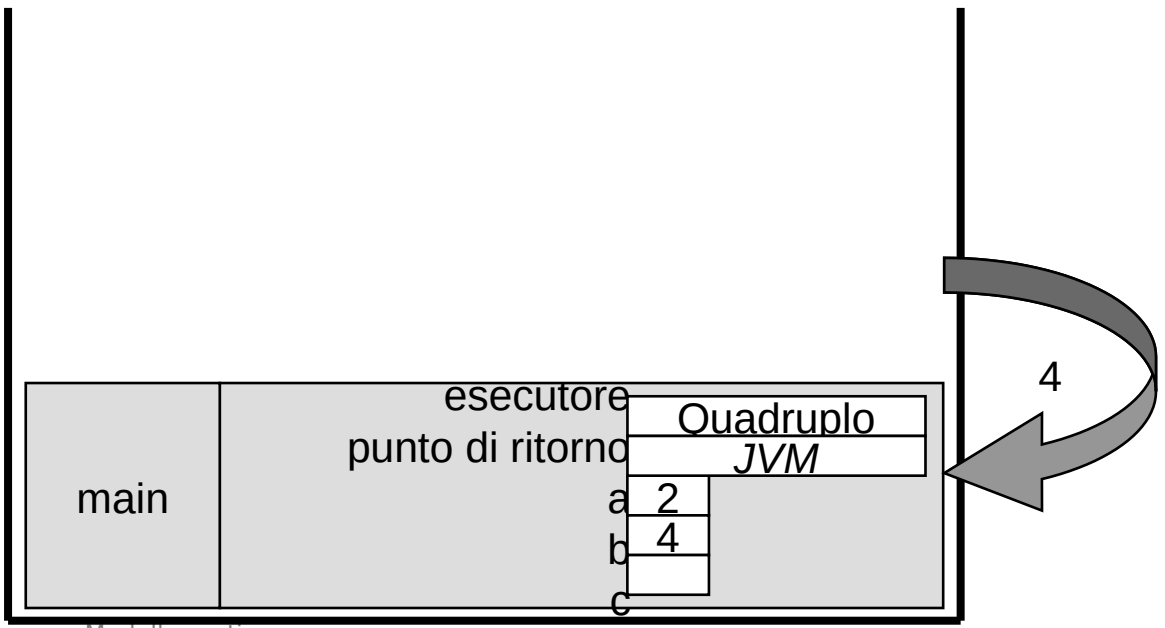


L’applicazione **Quadruplo**

```
class Quadruplo {
    public static int somma(int a, int b) {
        int c;                              // somma, 1
        c = a+b;                              // somma, 2
        return c;                             // somma, 3
    }
    public static int doppio(int n) {
        int d;                              // doppio, 1
        d = somma(n,n);                             // doppio, 2
        return d;                             // doppio, 3
    }
    public static void main(String[] args) {
        int a, b, c;                          // main, 1
        a = 2;                                // main, 2
        b = somma(a,a);   // b = 4             // main, 3
        c = doppio(b);    // c = 8             // main, 4
    }
}
```

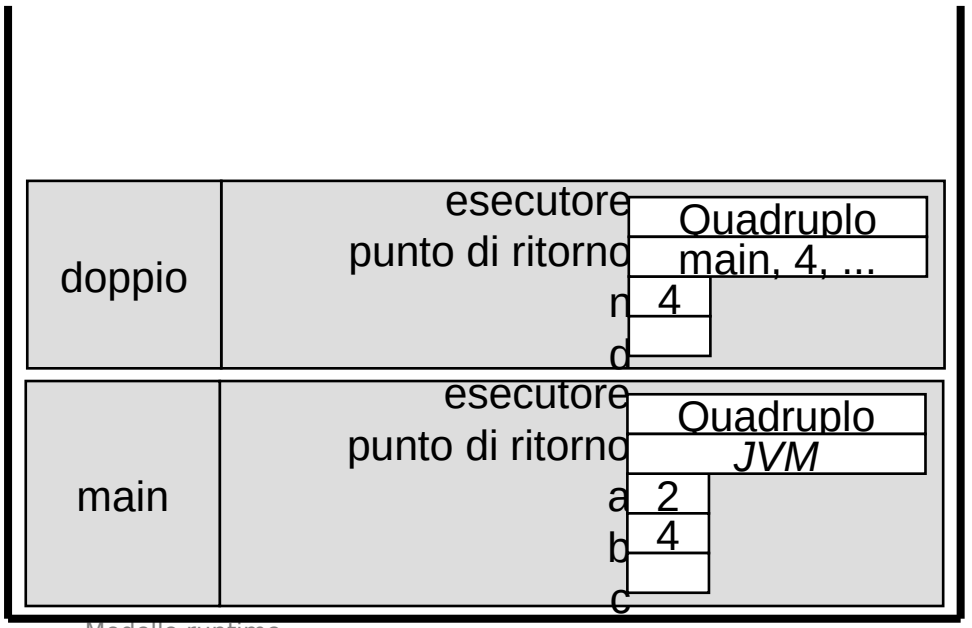
t=6 — assegnazione del valore restituito

Viene completata l’assegnazione alla variabile **b**



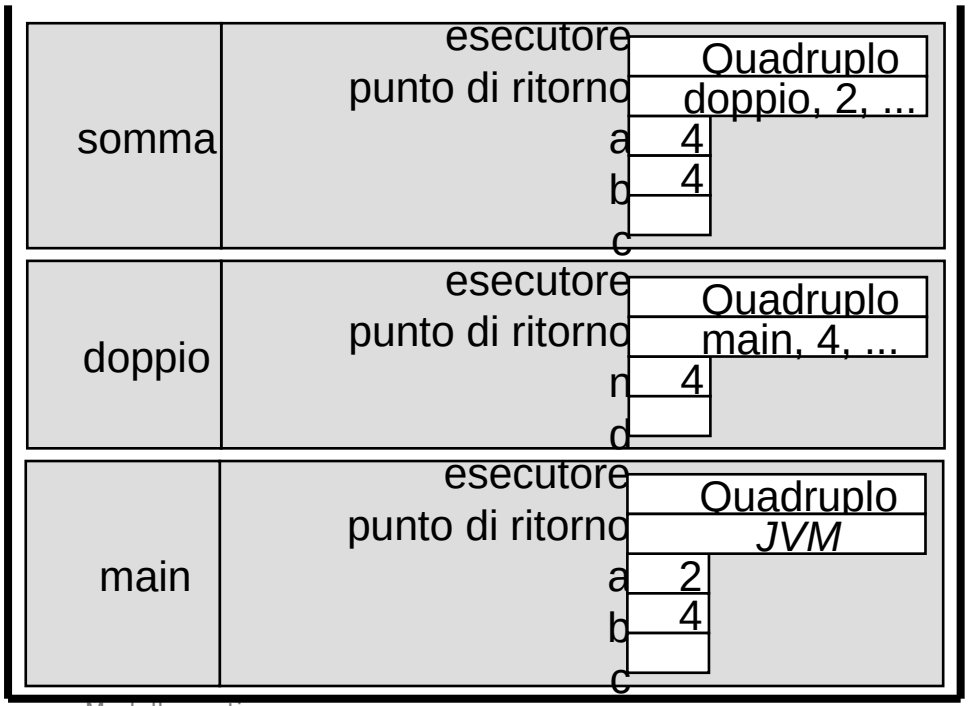
t=7 — invocazione e attivazione di doppio

Il metodo **main** invoca il metodo **doppio**



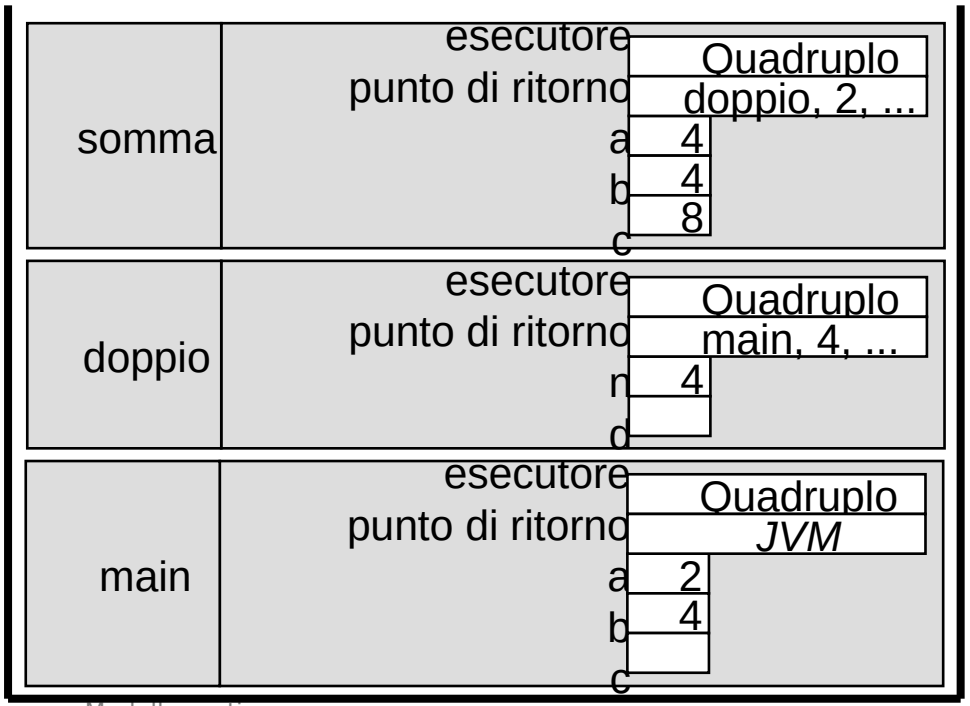
t=8 — invocazione e attivazione di somma

Il metodo **doppio** invoca il metodo **somma**



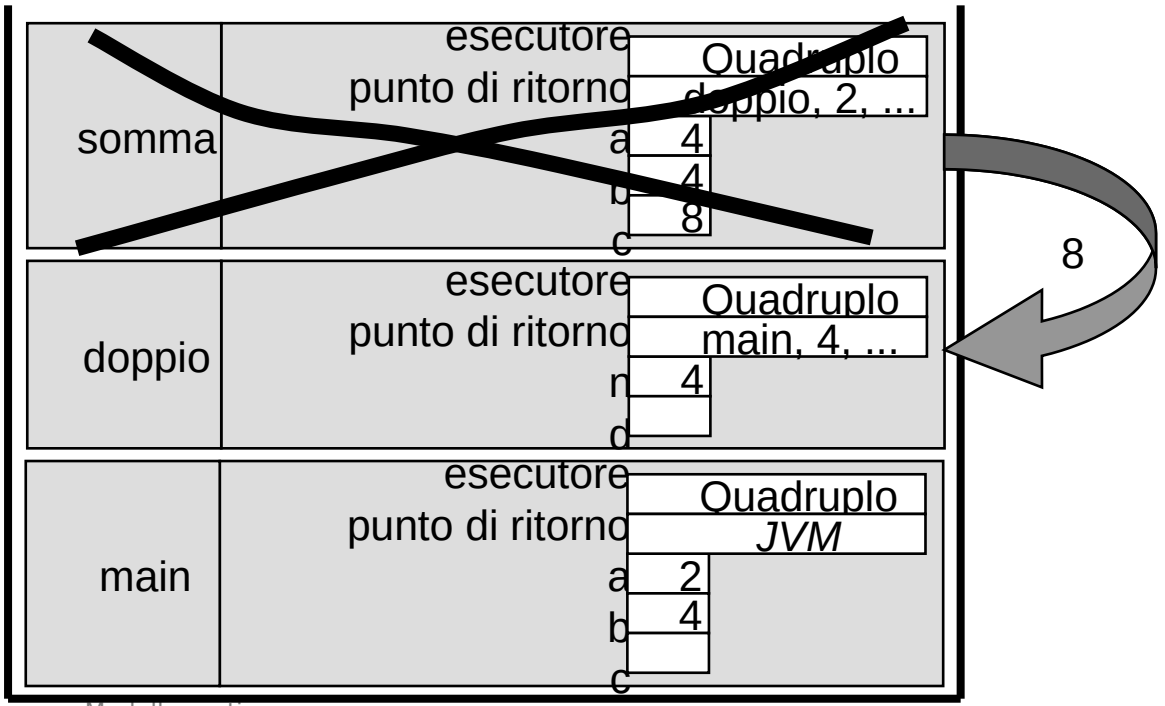
t=9 — esecuzione di una assegnazione

Viene eseguita l'assegnazione **c=a+b**



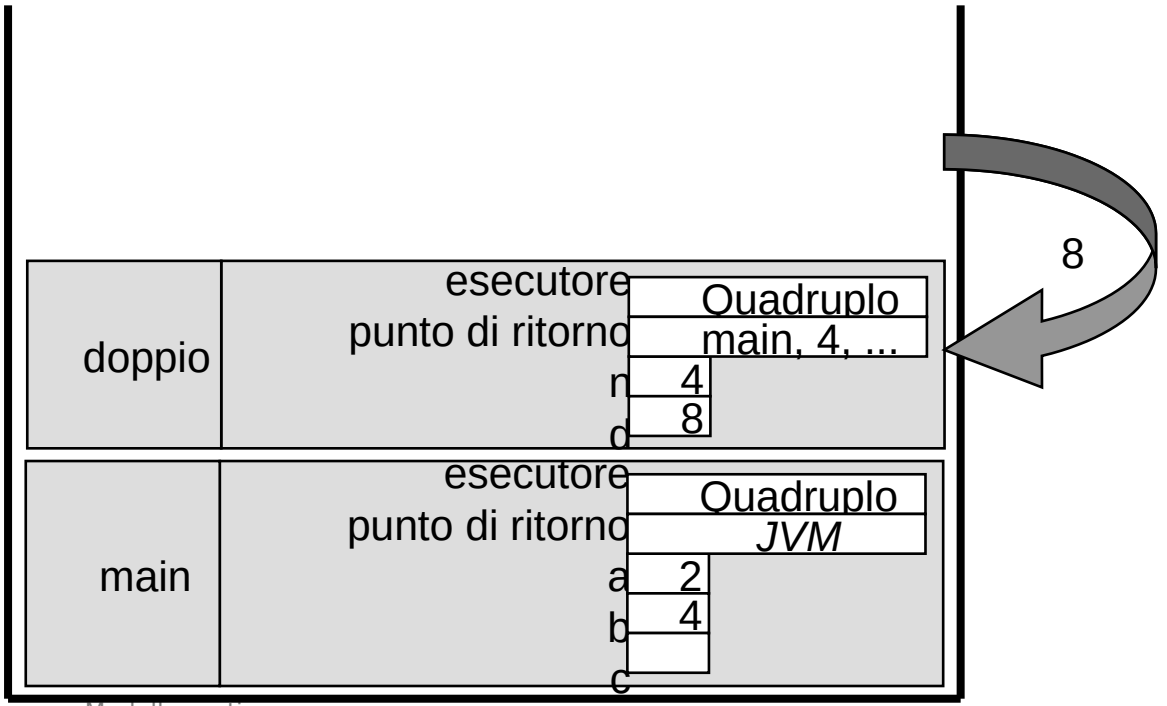
t=10 — terminazione di somma

Viene eseguita l’istruzione **return c** di **somma**



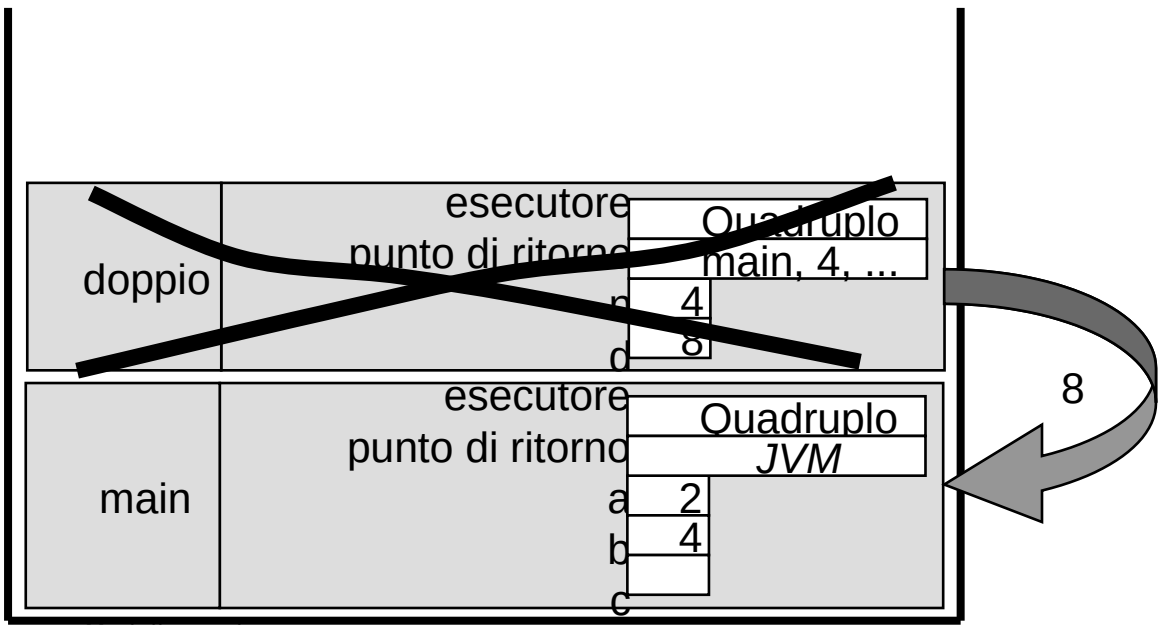
t=11 — assegnazione del valore restituito

Viene completata l’assegnazione alla variabile **d**



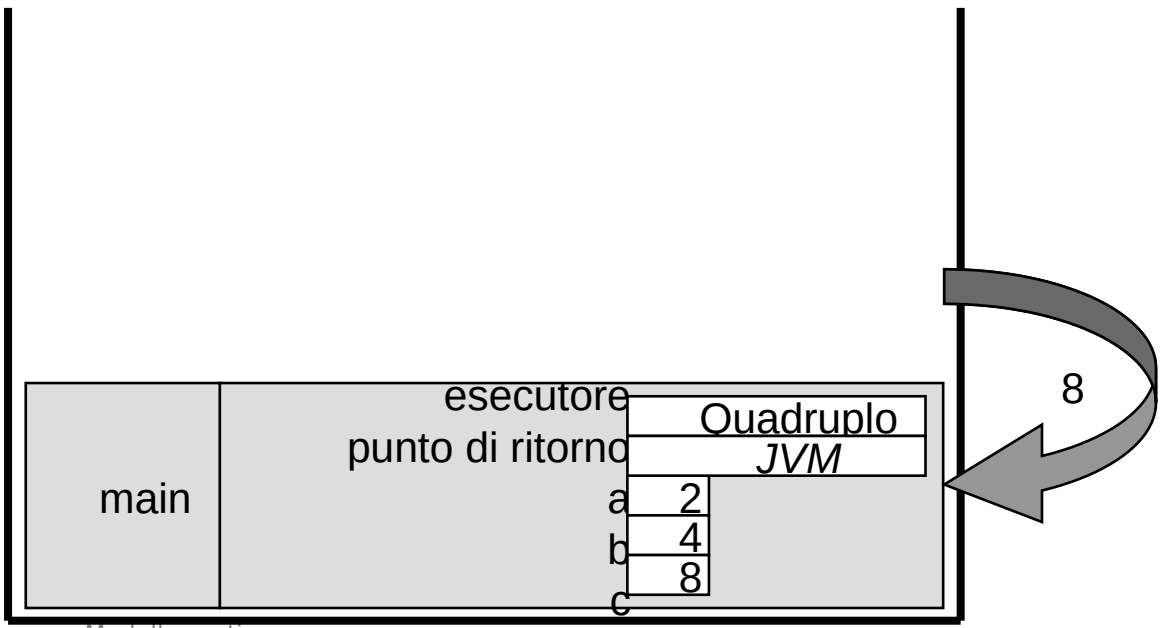
t=12 — terminazione di doppio

Viene eseguita l’istruzione **return d** di **doppio**



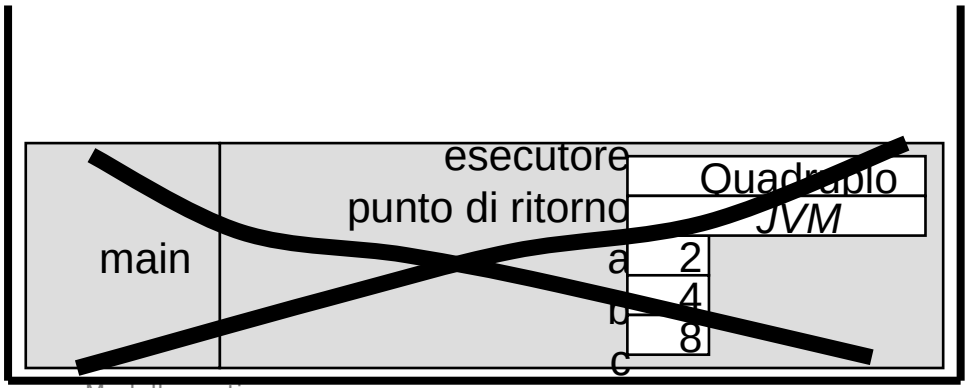
t=13 — assegnazione del valore restituito

Viene completata l’assegnazione alla variabile **c**



t=14 — terminazione di main

Termina anche l’esecuzione del metodo **main**



Esecuzione di metodi: discussione

Osservazioni

- i record di attivazione sono relativi alle attivazioni dei metodi
- ordine nell'allocazione/deallocazione di record di attivazione
- in questo esempio è possibile pensare a una gestione statica della memoria
  - in generale non è possibile

Esercitazioni (Linguaggi di programmazione quali Python)

- Esistono numerosi ambienti dove poter effettuare applicazioni
- Ed esempi di come si progetta e si gestisce il codice
- Esempi su youtube (basta digitare python in youtube, e si trovano tutorial semplici)
- https://www.youtube.com/watch?v=WK03M1ATxC0
- https://www.youtube.com/watch?v=XHzDHJ-BgvU

Compilatori ed ambienti

- Il compilatore di un linguaggio di programmazione o di un ambiente di sviluppo fornisce gli strumenti per ospitare un codice, tradurlo in linguaggio macchina e renderlo operativo
- Per semplificare esistono sistemi cloud (come per i documenti su google drive o su altro cloud, come overleaf...

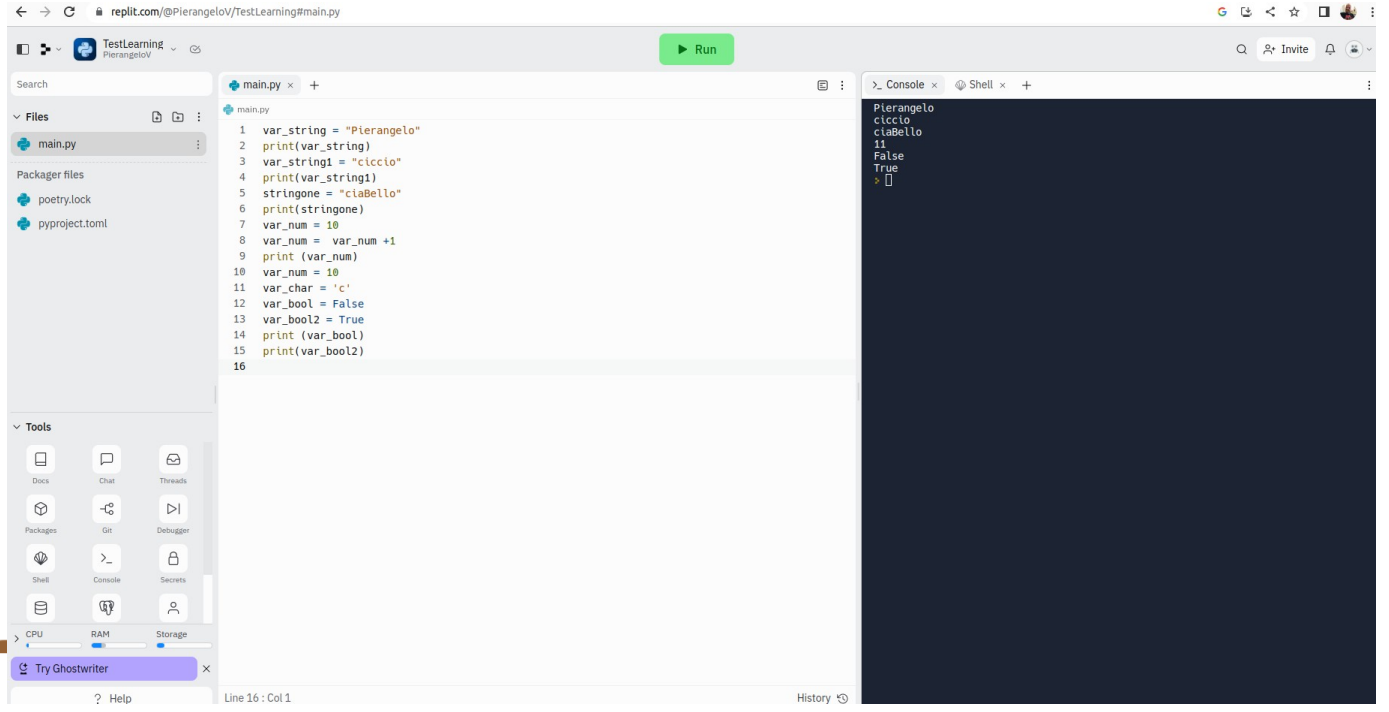
Esempio di variabili e di controllo

- https://replit.com/@PierangeloV/TestLearning#main.py

```
var_num=3;
if(var_num==3):
  print("uguale 3")
else:
  print("non e' uguale 3")
```

- Concetto di variabile e di controllo. Riferimenti alle strutture a blocchi
- Concetto di tipi di dati meno rigoroso da linguaggi standard

- Sistemi cloud based per la compilazione e definizione di spazio di lavoro e di coding
- Senza installare compilatori
- https://replit.com
-



Come definire un esempio per lavorare in modo semplice

- Un linguaggio di programmazione ha bisogno di
  - Sintassi, semantica, un compilatore, un traduttore e un esecutore
- Definire uno spazio di lavoro in cloud: un esempio
  - Replit.com; definire un account con un nome e una mail
  - Avviare un nuovo spazio di lavoro,
  - Scegliere l’ambiente Python
  - Scrivere il codice nel main.py
  - Se si scrive un altro file, definire import Nomefile.py nel main

```
24  var_num= 3
25  var_num-=2
26  if (var_num<=5 and var_num>2):
27    print("Variabile minore uguale di 5 e maggiore 2")
28  elif (var_num>2):
29    printf("Var magg 2")
30  else:
31    print("fine")
```

Ancora richiami (rapidi) di Python

- Nel main.py si puo' usare "import esMatrici"

esMatrici.py

```
1  vettore_v = [1,12,-23]
2  palazzo_v = ["mario", "luca", "giovanni", "andrea"]
3  matrice_m = [[12, 11, 10],[-1, 22, 30 ], [7,12,15]]
4  # m e' una matrice
5  # quadrata
6  print(vettore_v[0])
7  #scrivo il valore nelal prima posizione
8  print(palazzo_v[1])
9  print (matrice_m[0][1])
```

Cicli

- Sempre in reply.it. Il for esegue n con n il numero di valori che stanno nel vettore

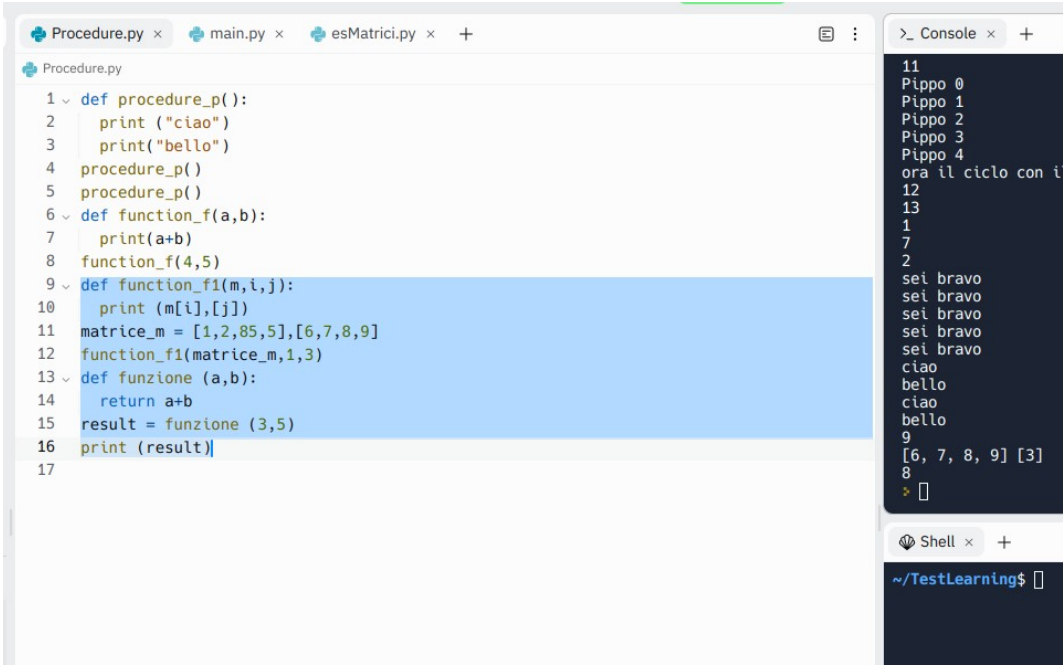
```
1  counter = 0
2  while (counter<5):
3    print("Pippo", counter)
4    counter=counter+1
5  print ("ora il ciclo con il for")
6  vettore_v = [12,13,1,7,2]
7  for n in vettore_v:
8    print(n)
9  for n in vettore_v:
10   print("sei bravo")
11
```

```
la variabile bella e' Pierangelo
fine
1
luca
11
Pippo 0
Pippo 1
Pippo 2
Pippo 3
Pippo 4
ora il ciclo con il for
12
13
1
7
2
sei bravo
sei bravo
sei bravo
sei bravo
sei bravo
```

Shell

Chiamate a procedure

- def function_f1(m,i,j):
- print (m[i],[j])
- matrice_m = [1,2,85,5],[6,7,8,9]
- function_f1(matrice_m,1,3)
- def funzione (a,b):
- return a+b
- result = funzione (3,5)
- print (result)



Classi ed esempi di programmazione a oggetti

```
class Persona:
  def __init__(self, a, b, c):
    self.nome = a
    self.cognome = b
    self.numeroAsn = c

p1 = Persona("Pierangelo", "Veltri", 123456789)
p2 = Persona("Luca", "Rossi", 102304050606)
print (p1.nome, p1.cognome, p1.numeroAsn)
print (p2.nome, p2.cognome, p2.numeroAsn)
```

Un esempio di definizione di classe Paziente

Sempre su

replit.com

EsempiClassi.py × main.py × esMatrici.py ×

EsempiClassi.py

```
1  class Persona:
2    def __init__(self, a, b, c):
3      self.nome = a
4      self.cognome = b
5      self.numeroAsn = c
6
7  class Paziente:
8    def __init__(self,a,b,c,d):
9      self.Nome = a
10     self.Cognome = b
11     self.MedicoDiBaseId = c
12     self.NumeroIdPz = d
13   def whoAmI(self):
14     print("Il Nome del Paziente è ", self.Nome)
15     print("Il Cognome del Paziente è ", self.Cognome)
16     print("Il numeroIdentificativo Ospedaliero del Paziente e' ",
   self.MedicoDiBaseId)
17     print ("Il Medico del Paziente ha il seguente Id", self.NumeroIdPz)
18 P1 = Paziente ("Pierangelo", "Veltri", 125, 999)
19 P1.whoAmI()
20
21 p1 = Persona("Pierangelo", "Veltri", 123456789)
22 p2 = Persona("Luca", "Rossi", 102304050606)
23 print (p1.nome, p1.cognome, p1.numeroAsn)
24 print (p2.nome, p2.cognome, p2.numeroAsn)
```

Console ×

```
ora il ciclo con il for
12
13
1
7
2
sei bravo
sei bravo
sei bravo
sei bravo
sei bravo
ciao
bello
ciao
bello
9
Il Nome del Paziente è  Pierangelo
Il Cognome del Paziente è  Veltri
Il numeroIdentificativo Ospedaliero del Paziente e'
Il Medico del Paziente ha il seguente Id 999
Pierangelo Veltri 123456789
Luca Rossi 102304050606
>
```

Shell ×

```
~/TestLearning$
```

Un esempio di definizione di classe Physiological Pump

- Nell’ambito della gestione dei dispositivi, un possibile dispositivo e’ la pompa insulinica.
- Viene comandata da un controllo logico che ne stabilisce lo stato e l’erogazione dei valori
- Il glucosimetro misura i valori
-

Definire il microcontrollore della pompa:

```
25
26  class PhisyoPump:
27    def __init__ (self, stato, id, value):
28      #status on/off, id identificativo, value Pressione
29      self.status = stato
30      self.ident = id
31      self.value = value
32    def whoAmI(self):
33      print(self.ident)
34      print (self.status)
35      print (self.value)
36    def setMyStatus (self,s):
37      self.status = s
38    def setMyValue (self,val):
39      self.value = val
40  new = PhisyoPump(1,2,3)
41  newPump = PhisyoPump("spento", 123, 0)
42  #ho creato una pompa con un valore spento
43  newPump.whoAmI()
44  newPump.setMyStatus("Acceso")
45  newPump.setMyValue(12)
46  newPump.whoAmI()
```

```
sei bravo
sei bravo
sei bravo
sei bravo
sei bravo
ciao
bello
ciao
bello
9
Il Nome del Paziente è  Pierangelo
Il Cognome del Paziente è  Veltri
Il numeroIdentificativo Ospedaliero del Paziente e'
Il Medico del Paziente ha il seguente Id 999
Pierangelo Veltri 123456789
Luca Rossi 102304050606
123
spento
0
123
Acceso
```

Shell × +

```
~/TestLearning$
```