

Programmazione e Algoritmi 1 - Prova pratica

A.A. 2025/26 — Compito A del 5/2/2026

prof. Gianluca Amato

Esercizio 1 (8 punti)

Scrivere una programma che prende in input da terminale due numeri interi n ed m , Utilizzando la libreria `ezgraphics`, il programma disegna una griglia formata da $n \times m$ quadrati di lato 40 pixel. Al centro di ogni quadrato deve essere scritto un numero progressivo che parte da 1 e si incrementa di 1 ad ogni quadrato, secondo l'ordine consueto di lettura.

Ad esempio, se l'utente immette i numeri 3 e 4 per, rispettivamente, il numero di righe e di colonne, il programma dovrà visualizzare la seguente griglia:

1	2	3	4
5	6	7	8
9	10	11	12

Successivamente, scrivete una seconda versione del programma che legge i numeri n ed m da un file di testo. Inoltre, anche i valori da inserire nella tabella devono essere letti da un file di testo (deve essere accettata una qualunque sequenza di numeri). Potete decidere voi il formato di questo file.

Esercizio 2 (6 punti)

Scrivere una funzione `estrai_maschera(data, maschera)`, dove `data` è una lista di oggetti qualunque e `maschera` è una lista di booleani. La funzione, senza modificare i parametri in input, restituisce una nuova lista che contiene gli elementi di `data` che corrispondono a valori `True` in `maschera`. Ad esempio `estrai_maschera(["a", "z", 5, 7], [True, False, False, True])` restituisce `["a", 7]`.

Se la lunghezza di `maschera` è minore di quella di `data`, gli elementi mancanti vengono considerati uguali a `False`. Se la lunghezza di `maschera` è maggiore di quella di `data`, gli elementi in più vengono semplicemente ignorati.

Successivamente, scrivere una variante della funzione chiamata `estrai_maschera_inplace(data, maschera)` che modifica direttamente la lista `data` invece di costruire una lista nuova.

Esercizio 3 (3 punti)

Scrivere due test nel framework `pytest` per verificare il corretto funzionamento di `estrai_maschera`. In particolare, si implementino due test.

- un test per le seguenti chiamate di funzione:
 - `estrai_maschera(["a", "z", 5, 7], [True, False, False, True])`
 - `estrai_maschera(["a", "z", 5, 7], [True])`
 - `estrai_maschera(["a", "z"], [False, True, True])`
- un test analogo ma per la funzione `estrai_maschera_inplace`.

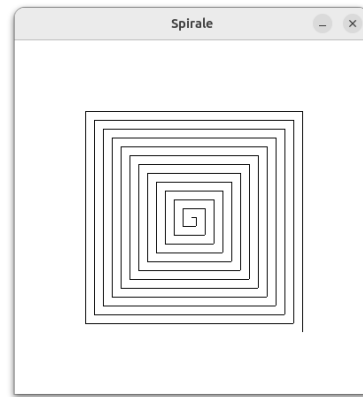
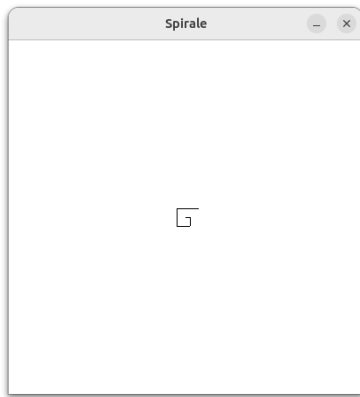
Programmazione e Algoritmi 1 - Prova pratica

A.A. 2025/26 — Compito B del 5/2/2026

prof. Gianluca Amato

Esercizio 1 (8 punti)

Scrivere una programma che visualizza una spirale quadrata usando la libreria `ezgraphics`. Il programma chiede all'utente il numero di linee di cui deve essere composta la spirale, quindi la visualizza su una finestra grafica. Questo è un esempio di output per una spirale di 5 linee e una di 50.



Inoltre, il programma deve salvare nel file `coordinate.txt` tutte le coordinate dei vertici della spirale che avete disegnato: ogni riga del file deve essere composta da due numeri, separati da spazi, contenenti le coordinate x ed y di un vertice.

Suggerimenti. Mantenere in delle apposite variabili le seguenti informazioni: ultimo vertice della spirale, direzione dell'ultimo tratto della spirale, lunghezza dell'ultimo tratto. Ad ogni passo sulla base di queste informazioni si traccia un segmento della spirale e si aggiornano le variabili.

Esercizio 2 (5 punti)

Scrivere la funzione `somma_opposti` che prende come parametro una lista di numeri interi e restituisce **True** se le somme del primo e ultimo elemento, del secondo e del penultimo, del terzo e del terzultimo, e così, via sono tutte uguali. In caso contrario la funzione restituisce **False**. Ecco alcuni esempi:

- `somma_opposti([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])` restituisce **True** perché $1 + 10 = 2 + 9 = 3 + 8 = 4 + 7 = 5 + 6 = 11$;
- `somma_opposti([1, 1, 1])` restituisce **True** perché il primo e ultimo elementi sono due numeri uno, che sommati fanno 2; analogamente, anche il secondo e il penultimo elemento sono due numeri uno (sebbene sia lo stesso elemento della lista) e sommati fanno anche loro 2;
- `somma_opposti([1, 2, 2, 10])` restituisce **False** perché $1 + 10 \neq 2 + 2$.
- `somma_opposti([])` restituisce **True** perché non ci sono coppie di elementi da sommare.

Esercizio 3 (4 punti)

Scrivere alcuni test nel framework `pytest` per verificare il corretto funzionamento di `somma_opposti`. In particolare, si deve controllare che il valore di ritorno sia corretto per:

- le liste di esempio dell'Esercizio 2;
- una lista generata casualmente in cui tutti gli elementi sono uguali: devono essere generati casualmente sia l'elemento da inserire nella lista sia la lunghezza della lista (entrambi numeri interi nell'intervallo da 0 a 99).

Programmazione e Algoritmi 1 - Prova pratica

A.A. 2025/26 — Compito C del 5/2/2026

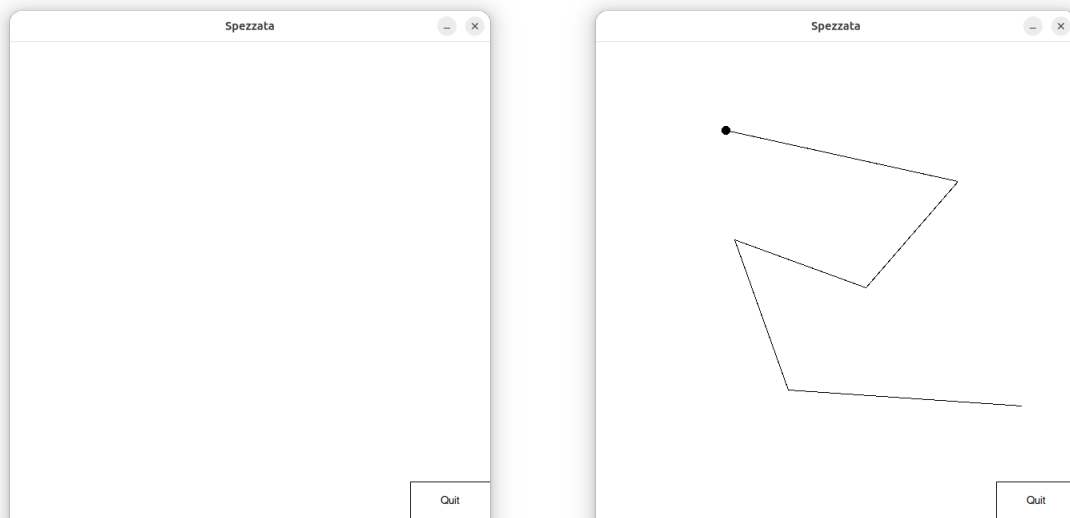
prof. Gianluca Amato

Esercizio 1 (8 punti)

Scrivere un programma che mostra una finestra completamente bianca di dimensione 600x600 pixel, tranne un piccolo rettangolo in basso a destra con la scritta “Quit”. Il programma disegna una spezzata, ovvero una sequenza di segmenti, messi in modo tale che un segmento inizia dove finisce il precedente. In pratica, ad ogni click del mouse il programma traccia un nuovo segmento, che va dalla fine del segmento precedente al punto dove si è cliccato. Ci sono solo due eccezioni a questa regola:

1. Il primo click del mouse non disegna alcun segmento, ma un cerchio nero di raggio 10 pixel centrato nel punto del click. Questo sarà anche il punto di partenza per il primo segmento.
2. Quando si clicca sul rettangolo in basso a destra il programma termina.

Trovate qui sotto due screenshot: in quello a sinistra il programma è appena partito, mentre in quello a destra il mouse è stato cliccato in 6 punti diversi, e il programma ha disegnato i segmenti che collegano questi punti.



Quando il programma termina, salva nel file `disegno.txt` le coordinate di tutti i punti che sono stati cliccati, in ordine di apparizione. Ogni riga deve contenere le coordinate di un punto, prima la coordinata x e poi la coordinata y, separate da una virgola.

Suggerimenti. Al solito, per acquisire la posizione in cui è avvenuto il click potete usare il metodo `getMouse()` della classe `GraphicsWindow`.

Esercizio 2 (6 punti)

Scrivere una funzione `add_total` che prende come parametro una matrice di numeri, anche frastagliata, e la modifica sul posto aggiungendo una nuova colonna con i totali delle righe. Ad esempio, se

```
m = [  
    [1, 2, 3],  
    [4, 5]  
]
```

dopo la chiamata a `add_total(m)` si avrà

```
m == [  
    [1, 2, 3, 6],  
    [4, 5, 9]  
]
```

Scrivere quindi una funzione `add_total2` che è simile ad `add_total` ma restituisce una nuova matrice invece di modificare quella passata come parametro.

Esercizio 3 (3 punti)

Utilizzare il framework `pytest` per collaudare le funzioni `add_total` e `add_total2` dell'esercizio precedente. Si considerino in particolare, per entrambe le funzioni, i casi corrispondenti ai seguenti parametri in input:

- l'esempio fornito nel testo dell'esercizio;
- una matrice con una sola riga e una sola colonna;
- una matrice senza nessuna riga.

Programmazione e Algoritmi 1 - Prova pratica

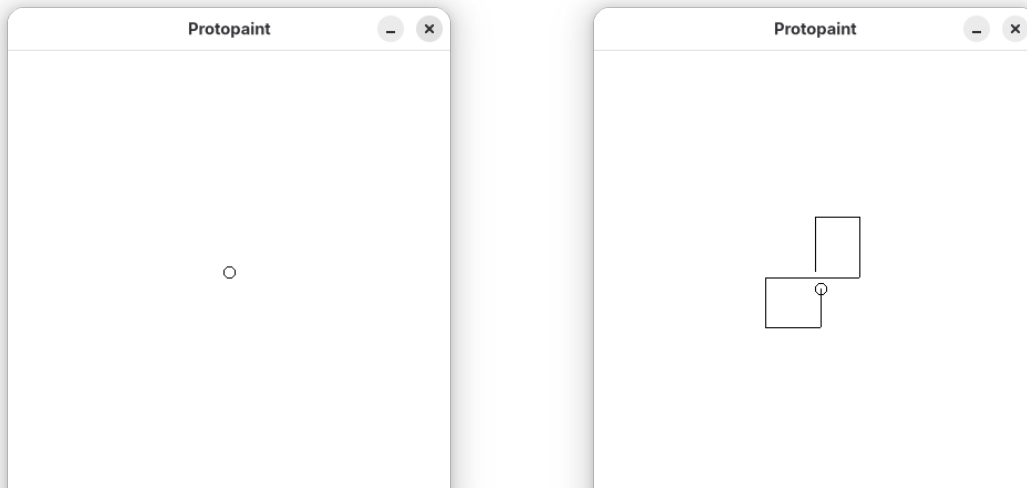
A.A. 2025/26 — Compito D del 5/2/2026

prof. Gianluca Amato

Esercizio 1 (8 punti)

Realizzare un programma di disegno rudimentale utilizzando la libreria **ezgraphics**. Il programma deve mostrare inizialmente una finestra completamente vuota, tranne per un “pennello” stilizzato (ad esempio, si può utilizzare un cerchio, ma altre scelte sono possibili). Con i tasti cursore è possibile spostare liberamente il pennello sullo schermo. Quando l’utente preme il tasto *spazio*, il pennello entra in modalità disegno: mentre si sposta lascia una linea nera sullo schermo. Premendo di nuovo spazio il pennello esce dalla modalità disegno e torna a essere un semplice puntatore che non lascia tracce dove passa. Il programma termina quando l’utente preme il tasto *Esc*.

Quello che segue è un esempio di come potrebbe apparire il programma. Lo screenshot di sinistra è stato preso appena il programma è stato avviato, quello di destra dopo aver disegnato una specie di otto inclinato.



Suggerimenti. Per acquisire il tasto premuto dall’utente si può utilizzare il metodo `getKey()` della classe `GraphicsWindow`. Quando bisogna spostare il “pennello”, potete eliminare quello vecchio con il metodo `removeItem` della classe `GraphicsCanvas` e ridisegnarlo nella nuova posizione, oppure potete usare il metodo `shiftItem` per spostarlo direttamente.

Esercizio 2 (5 punti)

Scrivere una funzione `squares(n)` che restituisce la lista di tutti i quadrati perfetti minori o uguali ad n . Si può assumere che n sia un intero maggiore o uguale a 0. Ad esempio, `squares(30)` restituisce `[0, 1, 4, 9, 16, 25]`. Si ricorda che un quadrato perfetto è un numero del tipo x^2 per qualche intero x , ed infatti 0, 1, 4, 9, 16 e 25 sono rispettivamente 0^2 , 1^2 , 2^2 , 3^2 , 4^2 e 5^2 .

Esercizio 3 (4 punti)

Scrivere due test nel framework `pytest` per verificare il corretto funzionamento di `squares`. In particolare, si implementino due test.

- un test che verifichi se le seguenti chiamate di funzione restituiscono il risultato desiderato:
 - `squares(5)`
 - `squares(1)`
- un test per il caso in cui il numero n è un intero positivo o nullo generato casualmente. In questo caso, il test deve controllare che:
 - la lista restituita dalla funzione non sia vuota;
 - tutti i numeri della lista siano minori o uguali ad n .

Programmazione e Algoritmi 1 - Prova scritta

A.A. 2025/26 — Compito A del 5/2/2026

prof. Gianluca Amato

Esercizio 4 (7 punti)

Eseguire passo-passo il seguente codice Python, utilizzando l'apposito modulo:

```
1 def zip(l1, l2):
2     if len(l1) == 0 or len(l2) == 0:
3         return []
4     else:
5         l = zip(l1[1:], l2[1:])
6         return [(l1[0], l2[0])] + l
7
8 print(zip([1,2,3,4], ["a", "b", "c"]))
```

Esercizio 5 (6 punti)

Scrivere una funzione `join_strings(l, sep)` che prende come argomenti una lista di stringhe `l` e una stringa `sep` e restituisce una nuova stringa ottenuta concatenando tutti gli elementi di `l` tra di loro, separati dalla stringa `sep`. Ad esempio `join_strings(["io", "sono", "Batman"], ":")` deve restituire la stringa `"io:sono:batman"`.

Esercizio 6 (4 punti)

Si consideri il seguente frammento di codice in Python:

```
x = False
l = [x, "Ciao", False ]
l2 = [ 1 ]
x = 1
l[2] = 4
```

Illustrare con un disegno la struttura di heap e stack alla fine dell'esecuzione di questo frammento.

Programmazione e Algoritmi 1 - Prova scritta

A.A. 2025/26 — Compito B del 5/2/2026

prof. Gianluca Amato

Esercizio 4 (7 punti)

Eseguire passo-passo il seguente codice Python, utilizzando l'apposito modulo:

```
1 def zip(l1, l2):
2     if len(l1) == 0 or len(l2) == 0:
3         return []
4     else:
5         l = zip(l1[1:], l2[1:])
6         return [(l1[0], l2[0])] + l
7
8 print(zip(["x", "y", "z"], [0, 1, 2, 3]))
```

Esercizio 5 (6 punti)

Scrivere una funzione `join_strings(l, sep)` che prende come argomenti una lista di stringhe `l` e una stringa `sep` e restituisce una nuova stringa ottenuta concatenando tutti gli elementi di `l` tra di loro, separati dalla stringa `sep`. Ad esempio `join_strings(["0", "1", "20"], "*")` deve restituire la stringa `"0*1*20"`.

Esercizio 6 (4 punti)

Si consideri il seguente frammento di codice in Python:

```
y = 3
l0 = ["Ciao", y, False]
l = [ l0, l0 ]
y += 2
l0[2] = True
```

Illustrare con un disegno la struttura di heap e stack alla fine dell'esecuzione di questo frammento.

Programmazione e Algoritmi 1 - Prova scritta

A.A. 2025/26 — Compito C del 5/2/2026

prof. Gianluca Amato

Esercizio 4 (7 punti)

Eseguire passo-passo il seguente codice Python, utilizzando l'apposito modulo:

```
1 def zip(l1, l2):
2     if len(l1) == 0 or len(l2) == 0:
3         return []
4     else:
5         l = zip(l1[:-1], l2[:-1])
6         return [(l1[-1], l2[-1])] + l
7
8 print(zip([1, 2, 3, 4], ["a", "b", "c"]))
```

Esercizio 5 (6 punti)

Scrivere una funzione `join_symbols(sep, l)` che prende come argomenti una stringa `sep` e una lista di stringhe `l`. La funzione restituisce una nuova stringa ottenuta concatenando tutti gli elementi di `l` tra di loro, separati dalla stringa `sep`. Ad esempio `join_symbols("//", ["--", "++", "**"])` deve restituire la stringa `--//++//**`.

Esercizio 6 (4 punti)

Si consideri il seguente frammento di codice in Python:

```
z = 3
l0 = ["Ciao", z, False]
l = [ l0, l0, l0]
z = False
l0[2] = 6
```

Illustrare con un disegno la struttura di heap e stack alla fine dell'esecuzione di questo frammento.

Programmazione e Algoritmi 1 - Prova scritta

A.A. 2025/26 — Compito D del 5/2/2026

prof. Gianluca Amato

Esercizio 4 (7 punti)

Eseguire passo-passo il seguente codice Python, utilizzando l'apposito modulo:

```
1 def zip(l1, l2):
2     if len(l1) == 0 or len(l2) == 0:
3         return []
4     else:
5         l = zip(l1[:-1], l2[:-1])
6         return [(l1[-1], l2[-1])] + l
7
8 print(zip(["x", "y", "z"], [0, 1, 2, 3]))
```

Esercizio 5 (6 punti)

Scrivere una funzione `join_symbols(sep, ls)` che prende come argomenti una stringa `sep` e una lista di stringhe `ls`. La funzione restituisce una nuova stringa ottenuta concatenando tutti gli elementi di `ls` tra di loro, separati dalla stringa `sep`. Ad esempio `join_symbols("--", ["$", "%", "&"])` deve restituire la stringa `"$--%--&"`.

Esercizio 6 (4 punti)

Si consideri il seguente frammento di codice in Python:

```
x = 3
l = [x, "Ciao", False]
l2 = [1, 1]
x = 5
l[2] = True
```

Illustrare con un disegno la struttura di heap e stack alla fine dell'esecuzione di questo frammento.