Python
Parte 2: le strutture di controllo

Mattia Cozzi
cozzimattia@gmail.com

a.s. 2024/2025

Contenuti

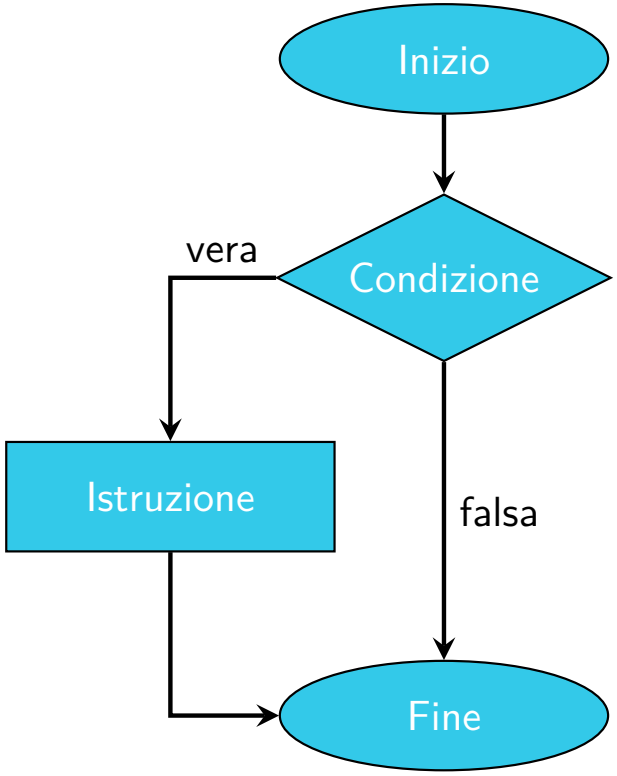
Strutture di controllo

Le istruzioni di un algoritmo possono:

- essere organizzate in sequenza, come abbiamo visto finora;
- presentare delle alternative (struttura condizionale o selezione);
- essere ripetute un certo numero di volte o finché si verifica una certa condizione (struttura iterativa o ciclo).

Ogni algoritmo può essere scritto con una combinazione di queste tre strutture fondamentali.

Selezione semplice



Un blocco di istruzioni viene eseguito quando la condizione è vera.

Selezione semplice in Python

La selezione semplice in linguaggio Python si ottiene con la sintassi:

```
1  x = 10
2  if x < 20:     #in questa riga ho un OPERATORE DI CONFRONTO
3    print("Il numero è piuttosto piccolo")  #riga INDENTATA
4  print("Fine dell\'algoritmo")
```

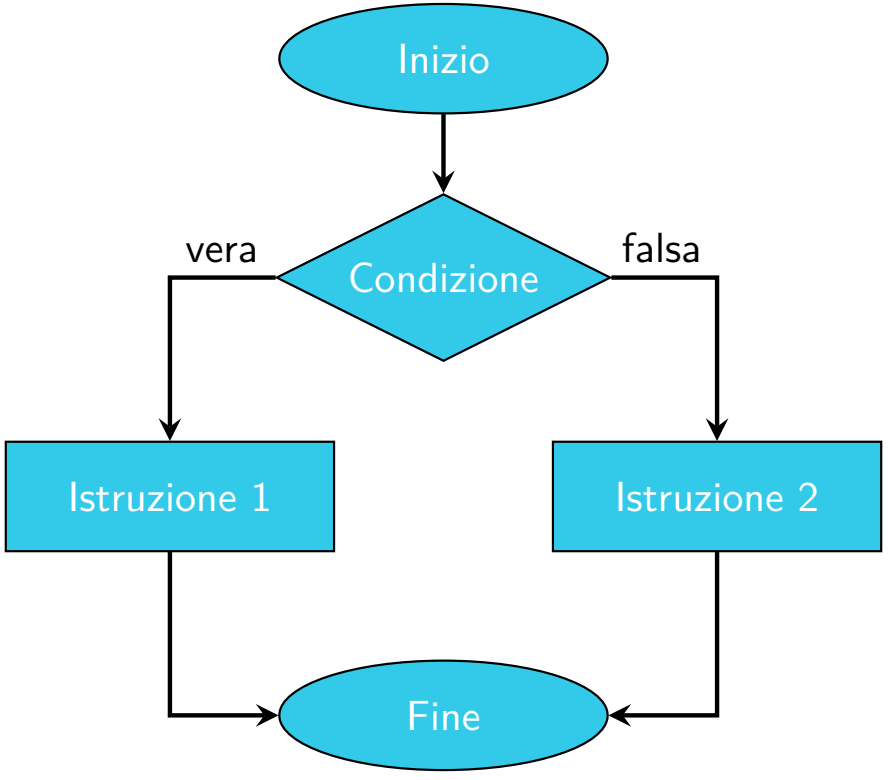
Il blocco di codice indentato viene eseguito solo a patto che la condizione dopo `if` risulti vera.

Attenzione anche ai due punti dopo la condizione.

Esercizi

1. Creare un algoritmo che chiede all'utente di inserire un numero e, se quel numero è multiplo di 3, mostra tale informazione a schermo.
2. Creare un algoritmo che chiede all'utente di inserire una parola complicata e, se quella parola è più lunga di 10 lettere, fa i complimenti all'utente per la sua scelta lessicale.

Selezione doppia



Un blocco di istruzioni viene eseguito a condizione vera, un altro blocco a condizione falsa.

Selezione doppia in Python

In Python:

```
1  x = 10
2  if x < 20:
3    print("Il numero è piuttosto piccolo")
4  else:
5    print("Il numero è grandicello")
6  print("Fine dell\'algoritmo")
```

Anche qui, attenzione ai due punti dopo **else**.

Esempio: stabilire se un numero è pari

Il seguente algoritmo chiede un numero all'utente ed esegue il casting da stringa a intero. Controlla poi se il resto della divisione tra il numero e due è zero. Se sì, il numero è pari, altrimenti è dispari.

```
1  numero = int(input("Inserisci un numero intero: "))
2  if (numero%2 == 0):   #attenzione al doppio uguale!
3    print("Il numero è pari!")
4  else:
5    print("Il numero è dispari!")
```

Esercizi

3. Creare un algoritmo che chiede all'utente di inserire una parola complicata e, se quella parola è più lunga di 10 lettere, fa i complimenti all'utente per la sua scelta lessicale. Se invece la parola è da 10 o meno lettere, deve spronare l'utente a far di meglio la prossima volta.
4. Scrivere un algoritmo che chieda all'utente di inserire il suo mese di nascita e il mese corrente e stabilisca poi, mostrandolo a schermo, se al mese corrente è più vicino il compleanno passato oppure il prossimo compleanno. Se i compleanni sono equidistanti, scegliere liberamente quale alternativa mostrare.

Selezioni annidate

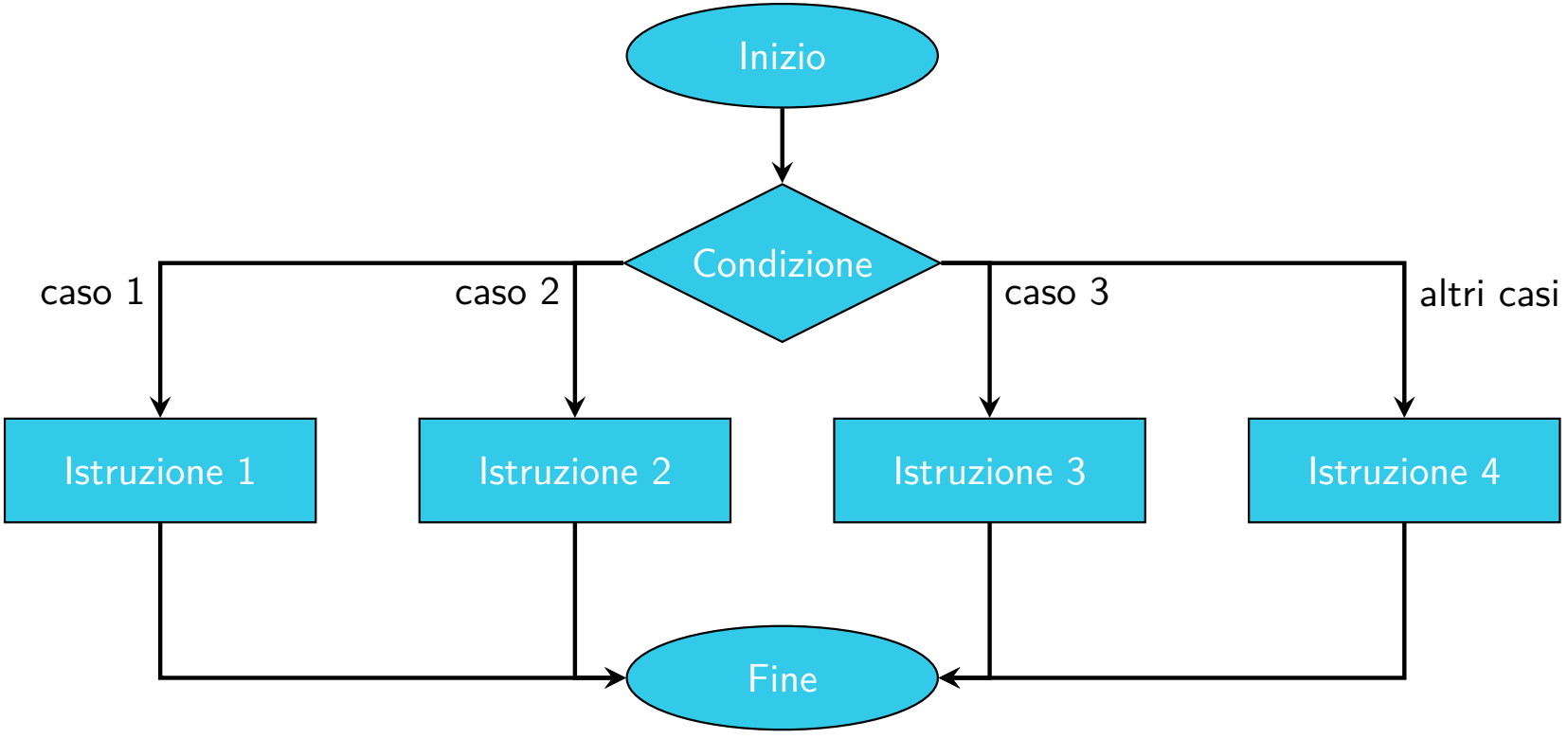
Per gestire casi complessi, possiamo anche inserire un **if** all'interno di un altro **if**, ottenendo una selezione annidata.

```
1  numero = int(input("Inserisci un numero pari minore di 20: "))
2  if (numero < 20):
3    if (numero%2 == 0):
4      print("Complimenti, il numero rispetta la richiesta")
5    else:
6      print("Il numero è abbastanza piccolo, ma non è pari")
7  else:
8    print("Hai inserito un numero troppo grande")
```

Esercizi

5. Usando una selezione annidata, creare un algoritmo che chieda all'utente di inserire i tre coefficienti di una equazione di secondo grado in forma canonica e che, se possibile, risolva l'equazione.

Selezione multipla



Istruzioni diverse vengono eseguite al verificarsi di certe condizioni. Se non se ne verifica nessuna, viene eseguite istruzioni ancora diverse.

Selezione multipla in Python

In Python:

```
1  voto = 18
2  if (voto < 18):
3    print("Esame non superato")          #primo caso
4  elif (voto == 18):
5    print("Esame superato a malapena")#secondo caso
6  elif (voto > 30):
7    print("Qualcosa non va")             #terzo caso
8  else:
9    print("Esame superato")              #tutto il resto
10 print("Fine dell\'algoritmo")
```

Le condizioni sono state messe tra parentesi tonde per aumentare la leggibilità del codice.

Anche qui, attenzione ai due punti dopo **elif** (che sta per *else if*).

Esercizi

6. In un cinema esistono 5 tipi di biglietti diversi. I bambini sotto i 6 anni pagano 3€, tra i 7 e i 12 anni pagano 5€, i ragazzi tra 13 e 18 anni pagano 7€, gli adulti pagano 10€ e infine gli anziani sopra i 70 anni entrano gratis. Scrivi un algoritmo che chieda all'utente la sua età e che mostri poi il corrispondente prezzo del biglietto.

Condizioni complesse

Le condizioni a cui attivare un `if` possono anche essere costruite con gli operatori logici visti in precedenza.

Ad esempio:

```
1  mese = int(input("Inserisci il tuo mese di nascita: "))
2  if (mese > 0) and (mese < 13):
3    print("Hai immesso un numero corretto per un mese")
4  else:
5    print("Il numero inserito non è un mese")
```

Un algoritmo equivalente al precedente:

```
1  mese = int(input("Inserisci il tuo mese di nascita: "))
2  if (mese < 1) or (mese > 12):
3    print("Il numero inserito non è un mese")
4  else:
5    print("Hai immesso un numero corretto per un mese")
```

Esercizi

7. Scrivere un algoritmo che chieda all'utente di inserire il suo nome e che controlli, mostrandolo a schermo, se il nome inizia per vocale.
8. Come l'esercizio precedente, ma controllando se il nome inizia e termina per vocale.
9. Creare un algoritmo che chieda all'utente di inserire i tre coefficienti di una disequazione di secondo grado in forma canonica e che, se possibile, risolva la disequazione.

Selezione breve

Esiste anche un costrutto molto rapido per delle selezioni doppie molto semplici, cioè selezioni in cui si esegue una sola istruzione se la condizione è vera.

Sintassi:

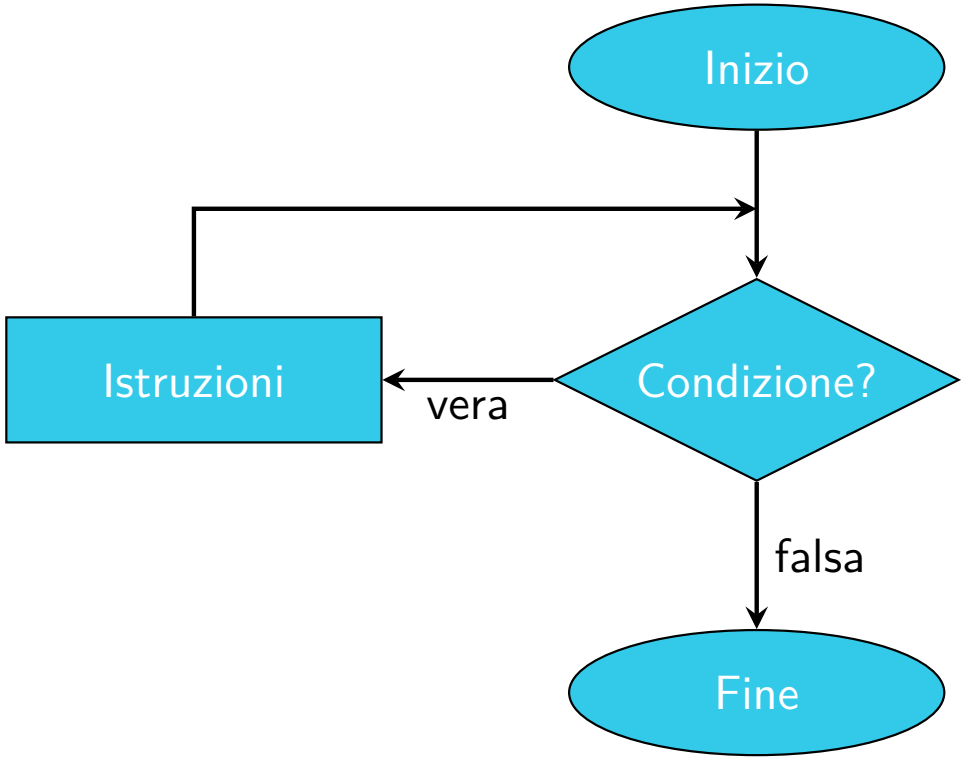
```
1  x = 6
2  y = 10
3  if x>y: print("Il primo numero è maggiore")
```

Cicli in Python

In Python esistono diversi tipi di ciclo:

- ciclo a condizione iniziale (iterazione precondizionata), in cui una condizione viene controllata prima di eseguire un blocco di codice;
- ciclo a conteggio, in cui un blocco di istruzioni viene eseguito un certo numero di volte.

Ripetizione precondizionata



Si controlla una condizione: se è vera, si esegue qualcosa e si torna indietro. La condizione viene ricontrollata, eccetera. Si esce dal ciclo quando la condizione risulta falsa.

Ripetizione precondizionata in Python

Usiamo la seguente sintassi:

```
while condizione:   #condizione di ingresso nel blocco
  istruzioni        #vengono eseguite a condizione VERA
```

Attenzione! Un ciclo **while** usato male potrebbe portare ad un loop infinito!

È fondamentale avere qualche variabile il cui valore cambi ad ogni esecuzione del ciclo e che permetta di uscire da esso ad un certo punto.

Esempio di ciclo con contatore

```
1  numero = int(input("Di quale numero vuoi la tabellina? "))
2  i = 0              #creazione della variabile contatore
3  while (i < 10): #ingresso nel ciclo
4    print(numero * (i+1)) #stampa un elemento della tabellina
5    i = i+1               #aumenta di 1 il valore del contatore
6  print("Finito!")
```

L'aumento del valore di `i` a riga 5 può anche essere ottenuto con la sintassi breve:

```
5    i += 1               #aumenta di 1 il valore del contatore
```

break

Se voglio forzare l'uscita dal ciclo al verificarsi di qualche condizione, uso l'istruzione **break**:

```
1  i = 0
2  while i < 10:
3    print(i)
4    if i == 3:
5      break   #se i vale 3, esci dal ciclo
6    i += 1
```

continue

Se voglio saltare l'esecuzione di un ciclo al verificarsi di qualche condizione e proseguire con il ciclo successivo, uso l'istruzione `continue`:

```
1  i = 0
2  while i < 10:
3    i += 1
4    if i == 3:
5      continue   #se i vale 3, vai al prossimo ciclo
6    print(i)
```

`else` in un ciclo

Aggiungendo un `else` a fine ciclo, posso far eseguire qualche operazione una volta che il ciclo è terminato:

```
1  i = 0
2  while i < 6:
3    print(i)
4    i += 1
5  else:
6    print("Il contatore i non è più minore di 6!")
```

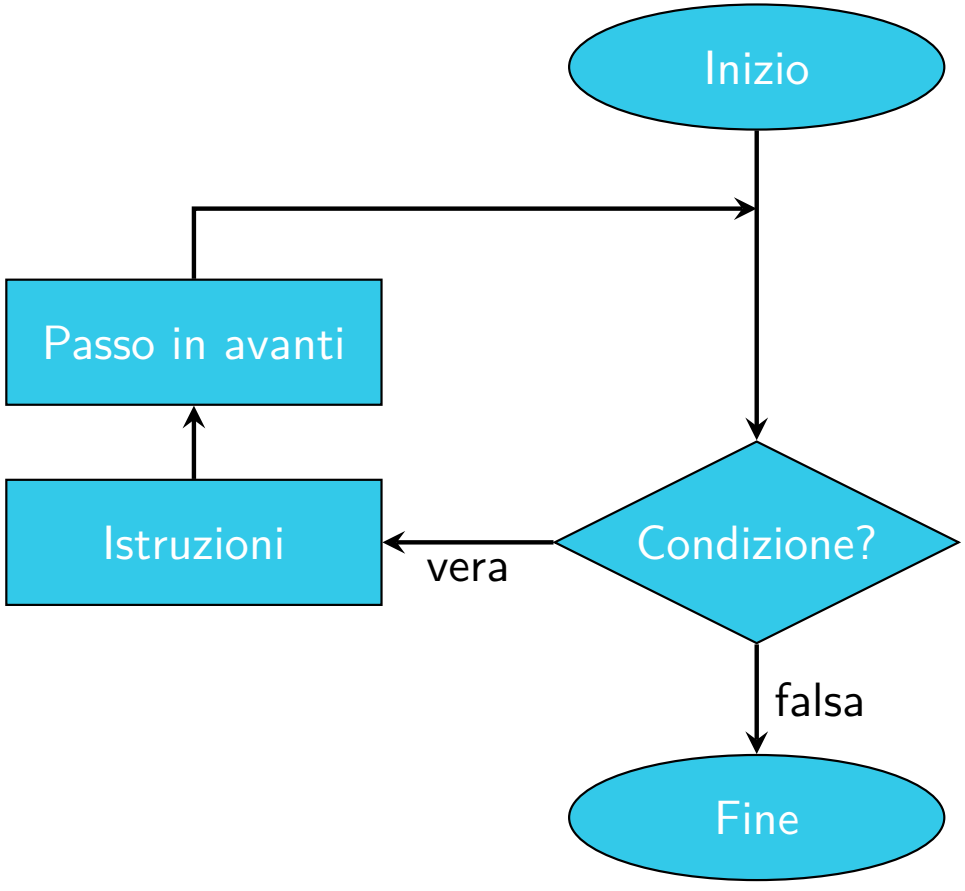
Esercizi

10. Scrivere un algoritmo che chieda un numero all'utente ed esegua la somma di tutti gli interi tra 0 e il numero inserito.
11. Scrivere un algoritmo che chieda un numero all'utente e calcoli il fattoriale di quel numero.
12. Scrivere un algoritmo che chieda numeri all'utente e li sommi. L'algoritmo deve proseguire fino a che l'utente non digita 0.
13. Scrivere un algoritmo che che chieda all'utente di quale numero voglia calcolare il quadrato, legga l'input dell'utente e ne mostri il quadrato. Successivamente, chiedere se si vuole calcolare un altro quadrato. Se l'utente risponde Y, allora chiedere un altro valore e ripetere il calcolo. Se l'utente risponde N, uscire dall'algoritmo.

Esercizi

14. In un cinema esistono 5 tipi di biglietti diversi. I bambini sotto i 6 anni pagano 3€, tra i 7 e i 12 anni pagano 5€, i ragazzi tra 13 e 18 anni pagano 7€, gli adulti pagano 10€ e infine gli anziani sopra i 70 anni entrano gratis. Scrivi un algoritmo che chieda quanti biglietti si vogliono acquistare e successivamente, per ogni biglietto, chieda l'età dell'acquirente e valuti quindi quanto si deve pagare. L'algoritmo deve infine mostrare il prezzo finale, cioè la somma di tutti i prezzi dei biglietti.

Ripetizione con contatore



Questo tipo di ciclo serve per iterare delle istruzioni all'interno di una sequenza di valori.

Ripetizione con contatore in Python

Spieghiamo un ciclo for con un esempio:

```
cities = ["Milano", "Roma", "Napoli"] #creazione di una lista
for x in cities:                      #scorre tutta la lista
  print(x)                            #e stampa il valore
```

Un altro esempio:

```
string = "endecasillabo" #creazione di una stringa
for x in string:         #scorre tutta la stringa
  print(x)               #e stampa il valore
```

La funzione `range()`

In Python per generare velocemente una lista di valori numerici possiamo usare la funzione `range(num)`, che crea una lista di numeri interi da `0` a `num-1`.

Esempio semplice di utilizzo, per stampare la tabellina del 7:

```
1  for x in range(10):
2    print(7 * (x+1))
```

Sintassi avanzata per `range()`:

```
1  #lista di valori da 2 a 10 (2 incluso, 10 escluso)
2  range(2, 10)
3
4  #lista di valori da 5 (incluso) a 100 (escluso) con passo 3
5  range(5, 100, 3)
```

break, continue, else

Come per il ciclo `while`, possiamo utilizzare le istruzioni `break`, `continue` ed `else` in un ciclo `for`.

```
1  frutti = ["Mela", "Banana", "Arancia", "Uva", "Pera", "Fragola",
   ↪  "Ananas", "Mango", "Kiwi", "Mirtillo"]
2  for x in frutti:
3    print(x)
4    if x == "Pera":
5      break
```

```
1  frutti = ["Mela", "Banana", "Arancia", "Uva", "Pera", "Fragola",
   ↪  "Ananas", "Mango", "Kiwi", "Mirtillo"]
2  for x in frutti:
3    if x == "Pera":
4      continue
5    print(x)
```

Esercizi

15. Scrivere un algoritmo che chieda all'utente un valore numerico e mostri la sequenza dei numeri di Fibonacci minori del numero inserito.
16. Usando la funzione `range()`, scrivere un algoritmo che chieda un numero all'utente e che mostri tutti i numeri interi da 0 a quel numero.
17. Scrivere un programma che chieda una stringa all'utente e ne faccia lo spelling.

Esercizi

18. Scrivere un algoritmo che chieda all'utente di inserire una parola e che la scriva successivamente al contrario (in un'unica stringa).
19. Scrivere un algoritmo che chieda all'utente di inserire una stringa, una lettera e un numero e che sostituisca poi nella stringa tutte le occorrenze della lettera con il numero.
20. Scrivere un algoritmo che chieda all'utente di inserire una parola e che controlli se tale parola è palindroma.
    Suggerimento: prima di eseguire il controllo, convertire tutta la parola in minuscolo.