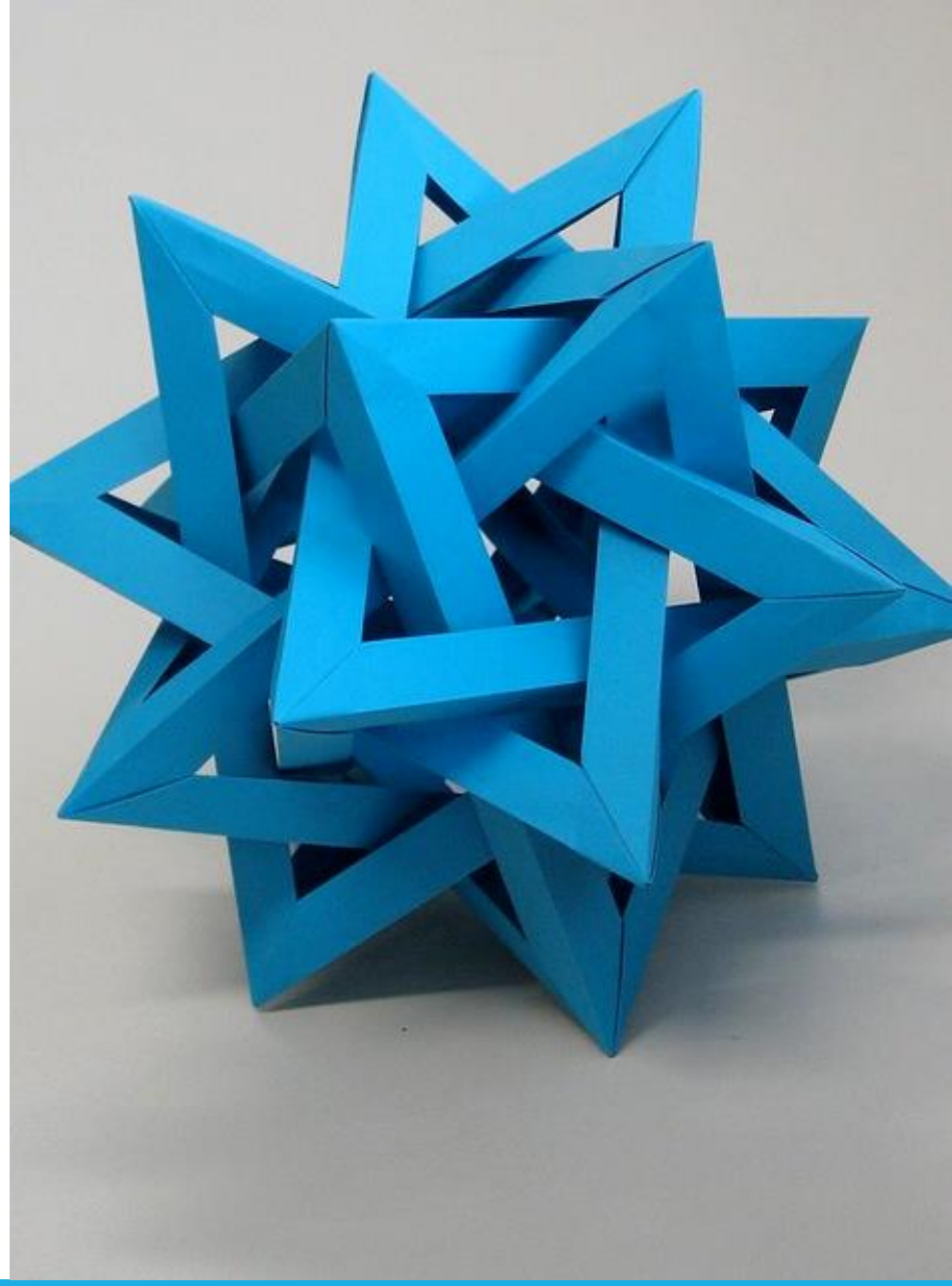
Politecnico di Torino

Dipartimento
di Automatica e Informatica

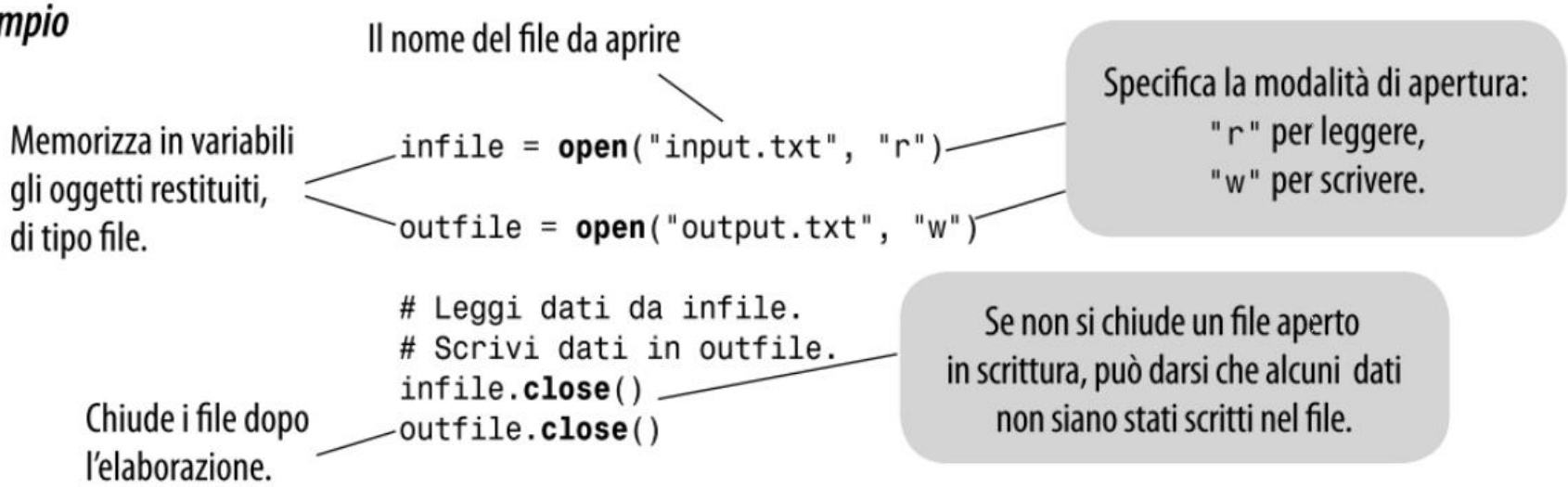
Laboratorio 10

APERTURA E SCRITTURA DI FILE, ELABORAZIONE DI DATI, GESTIONE ERRORI



Sintassi 7.1 Apertura e chiusura di file

Esempio



Esercizio 1

Esercizio 1. Scrivete un programma che legga il file di testo *input.txt*. Ogni riga letta va scritta nel file *output.txt* preceduta dal numero di riga, inserita come commento tra caratteri /* e */. Se, ad esempio, il file *input.txt* fosse il seguente:

Enola Gay
è il bombardiere che il 6 agosto 1945,
sganciò su Hiroshima la prima bomba atomica
soprannominata Little Boy.

Il file output.txt generato sarebbe:

/*1*/Enola Gay
/*2*/è il bombardiere che il 6 agosto 1945,
/*3*/sganciò su Hiroshima la prima bomba atomica
/*4*/soprannominata Little Boy.
[P7.2]

Esercizio 1 – *il codice Python*

```python
# Read the file names from the user.
inputName = 'input.txt'
outputName = 'output.txt'

# Open the input and output file names.
inf = open(inputName, "r")
outf = open(outputName, "w")

# Read lines from the file and save them to a new file with line numbers.
number = 1
for line in inf:
    outf.write("/* " + str(number) + " */ " + line)
    number = number + 1

# Close the files.
inf.close()
outf.close()
```

Esercizio 2

Esercizio 2. Scrivete un programma che legga tutte le righe di un file di testo (*input.txt*), ne inverta l'ordine e le scriva in un altro file (*output.txt*). Ad esempio, se il file *input.txt* contiene queste righe:

Mary had a little lamb
Its fleece was white as snow

And everywhere that Mary went
The lamb was sure to go.

allora il file *output.txt* conterrà:

The lamb was sure to go.
And everywhere that Mary went
Its fleece was white as snow
Mary had a little lamb

[P7.9]

Ricordare:

- Il metodo `readlines()` legge l'intero file coma una lista di stringhe (una stringa per ogni linea del file)

```
lines = infile.readlines()
```

```
# scorciatoia (non molto leggibile…)
lines = list(infile)
```

```
# equivalente a:
lines = []
for line in infile:
  lines.append(line)
```

Esercizio 2 – *il codice Python*

```
import sys

# Read the input file name from the command line and open the input file.
inputFile = 'input.txt'
inf = open(inputFile, "r")

# Read the lines from the input file.
lines = inf.readlines()

# Close the input file.
inf.close()

# Read the output file name from the command line and open the output file.
outputFile = 'output.txt'
outf = open(outputFile, "w")

# Write the lines in reverse order.
for i in range(len(lines) - 1, -1, -1):
    outf.write(lines[i])

# Close the output file.
outf.close()
```

Esercizio 3

Esercizio 3. Scrivere un programma che cerchi una data parola nel contenuto di un gruppo di file. Il programma innanzitutto chiede l'elenco dei file (da inserire su una sola riga, separati da virgole e la parola da cercare. I nomi dei file saranno memorizzati in una lista (files), la parola da cercare viene memorizzata in una variabile.
Occorre visualizzare tutte le righe che contengono la parola, indipendentemente da maiuscole o minuscole, ciascuna riga preceduta dal nome del file in cui si trova. Ad esempio, se la parola fosse "`ring`", e lista contenesse:

```
book.txt, address.txt, homework.py
```

allora il programma potrebbe visualizzare quanto segue:

```
book.txt: There is only one Lord of the Ring, only one who can bend
it to his will
book.txt: The ring has awoken; it's heard its masters call.
address.txt: Kris Kringle, North Pole
address.txt: Homer Simpson, Springfield
homework.py: string = "text"
```

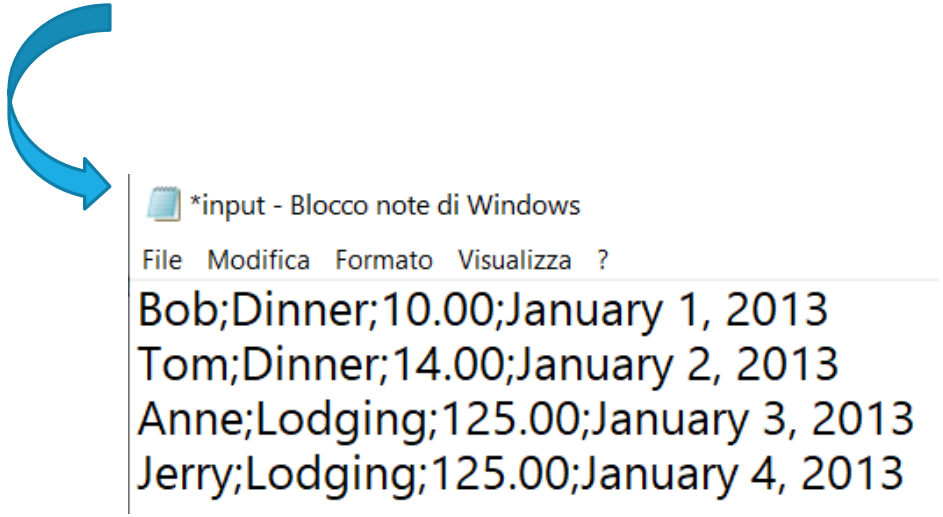
La parola da cercare viene memorizzata in una variabile e sono valide anche le parole che contengono quella da cercare (es. `ring` e `ringhiare`). [P7.6]

Esercizio 3 – *il codice Python*

```
import sys

# Target files
filenames = input('Insert the list of files to search (separate with ,): ')

# Extract the target wordfrom the command line.
target = input('Insert the word to search: ')

# For each filename on the command line.
for filename in filenames.split(','):
    # Save the filename and open the file.
    inf = open(filename.strip(), "r")

    # Search each line in the file for the target.
    for line in inf:
        if target in line:
            print("%s: %s" % (filename, line), end="")

    # Close the file.
    inf.close()
```

Esercizio 4

Esercizio 4. Il responsabile amministrativo di un albergo registra le vendite in un file di testo. Ogni riga contiene le seguenti 4 informazioni, separate da caratteri “punto e virgola”: il nome del cliente, il servizio venduto (ad esempio, cena, conferenza, alloggio, e così via), l’importo pagato e la data dell’evento. Scrivete un programma che legga un tale file di testo e visualizzi l’importo totale relativo a ciascun tipo di servizio, segnalando un errore se il file non esiste oppure se il suo formato non è corretto (verificando cioè che ci siano 4 campi per riga e che il prezzo sia `float`). [P7.29]



Esercizio 4

Sintassi

```
try :
   enunciati
   . . .
except TipoDiEccezione1 :
   enunciati
   . . .
except TipoDiEccezione2 as nomeVariabile :
   enunciati
   . . .
```

Esempio

```
try :
   infile = open("input.txt", "r")

   line = infile.readline()
   process(line)

except IOError :
   print("Could not open input file.")

except Exception as exceptObj :
   print("Error:", str(exceptObj))
```

Questa funzione può sollevare un'eccezione di tipo `IOError`.

Quando viene sollevata un'eccezione di tipo `IOError`, l'esecuzione riprende da qui.

Qui possono comparire ulteriori clausole `except`; le eccezioni più specifiche vanno elencate prima di quelle più generiche.

Questo è l'oggetto di tipo eccezione che è stato sollevato.

Esercizio 4 – *il codice Python*

```
# Read the filename from the user.
filename = input("Enter the name of the file to display: ")

# Open the file.
try:
    inf = open(filename, "r")
except IOError:
    exit("That file couldn't be opened.")

# Read the lines, adding new categories as they are found and keep track
# of the totals.
categories = []
totals = []
for line in inf:
    # Verify that the line has the required number of items.
    parts = line.split(";")
    if len(parts) != 4:
        exit("There is an invalid line in the file.")

    # Verify that the amount is a valid number.
    try:
        amount = float(parts[2])
    except ValueError:
        exit("There is an invalid number in the file.")

    # If the category already exists then add to the total, otherwise add a
    # new category.
    if parts[1] in categories:
        index = categories.index(parts[1])
        totals[index] = totals[index] + amount
    else:
        categories.append(parts[1])
        totals.append(amount)

# Close the file.
inf.close()

# Display the results.
print("Totals:")
for i in range(0, len(categories)):
    print("  %s: %.2f" % (categories[i], totals[i]))
```

Esercizio 5

Esercizio 5. *Cifratura monoalfabetica casuale.* Il cifrario di Cesare, che trasla ogni lettera di una quantità fissa, è troppo facile da violare. Ecco un'idea migliore: come chiave, invece di usare un numero, usiamo una parola, che immaginiamo essere FEATHER. Per prima cosa eliminiamo le lettere duplicate dalla parola chiave, ottenendo FEATHR, poi aggiungiamo in fondo ad essa le altre lettere dell'alfabeto, in ordine inverso:

F	E	A	T	H	R	Z	Y	X	W	V	U	S	Q	P	O	N	M	L	K	J	I	G	D	C	B

Ora cifriamo le lettere, seguendo questo schema:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
F	E	A	T	H	R	Z	Y	X	W	V	U	S	Q	P	O	N	M	L	K	J	I	G	D	C	B

Scrivete un programma che cifri o decifri un file di testo (`encrypted.txt`) usando una parola chiave, inserita dall'utente e memorizzata in una variabile, e scrivendo le informazioni decifrate nel file di testo `output.txt`. [P7.20]

Esercizio 5 – *il codice python*

```python
def main():
    keyword = input('Insert the keyword: ')
    keyword = remove_duplicates(keyword)
    keyword = keyword.upper()

    # Extract the filenames from the command line and open them.
    input_name = input('Input file name: ')
    inf = open(input_name, "r", encoding='utf-8')

    output_name = input('Output file name: ')
    outf = open(output_name, "w", encoding='utf-8')

    # Construct the full key string with the remaining letters of the
    # alphabet appended in reverse order.
    key_string = keyword
    for ch in "ZYXWVUTSRQPONMLKJIHGFEDCBA":
        if ch not in key_string:
            key_string = key_string + ch

    action = input('Do you want to encrypt or decrypt (e/d)? ')
```

```python
    # Encrypt the file.
    if action == 'e':
        for line in inf:
            for ch in line:
                pos = -1
                if "A" <= ch <= "Z":
                    pos = ord(ch) - ord("A")
                    outf.write(key_string[pos])
                elif "a" <= ch <= "z":
                    pos = ord(ch) - ord("a")
                    outf.write(key_string[pos].lower())
                else:
                    outf.write(ch)
    # Decrypt the file.
    elif action == 'd':
        for line in inf:
            for ch in line:
                out_ch = ch
                if ch in key_string:
                    out_ch = chr(ord("A") + key_string.index(ch))
                elif ch in key_string.lower():
                    out_ch = chr(ord("a") + key_string.lower().index(ch))
                outf.write(out_ch)
    # Handle the case where the user failed to specify the operation.
    else:
        exit("Either d or e must be provided for decrypt or encrypt.")

    # Close the files.
    inf.close()
    outf.close()
```

Esercizio 5 – *il codice python*

```
def remove_duplicates(s):
    """
    Create a new version of a string containing no duplicate letters.

    :param s: the string to remove duplicate letters from
    :return: a new string where all duplicate letters have been removed
    """
    retval = ""
    for ch in s:
        if ch not in retval:
            retval = retval + ch
    return retval


# Call the main function.
main()
```

Esercizio 6

Esercizio 7. Scrivere un programma che visualizzi l'elenco degli esami superati da uno studente, con i relativi voti. È disponibile un file, *classes.txt*, che contiene i nomi di tutti gli insegnamenti erogati nell'istituto scolastico (un college statunitense), il cui contenuto sarà simile a questo:

```
CSC1
CSC2
CSC46
CSC151
MTH121
…
```

Continua nella prossima slide…

Esercizio 6

Poi, per ogni insegnamento, è disponibile un file (il cui nome è pari al codice dell'insegnamento seguito da .txt) che elenca gli studenti che hanno superato il relativo esame e contiene i numeri identificativi degli studenti (ID) e i voti, come questo, che potrebbe essere il file *CSC2.txt*:

```
11234 A-
12547 B
16753 B+
21886 C
…
```

Scrivere un programma che chieda all'utente l'identificativo (ID) di uno studente e visualizzi l'elenco degli esami che tale studente ha superato, con i relativi voti ottenuti, come in questo esempio:

Esercizio 6 – *il codice python*

```
# Read the student id from the user.
sid = input("Enter the student id: ")

# Process each class found in classes.txt.
courses = open("classes.txt", "r")
print("Student ID", sid)
for course in courses:
    # Remove the trailing newline from the class number.
    course = course.rstrip()

    # Open the list for the class and search for the student.
    inf = open(course + ".txt", "r")
    for line in inf:
        parts = line.split()
        # If the student is in the class then print the mark.
        if sid == parts[0]:
            print(course+' '+line, end="")

    # Close the class file.
    inf.close()

# Close the list of classes.
courses.close()
```

Esercizio 7

Esercizio 6. *Cifrario di Playfair*. Un diverso modo per impedire la decifrazione di un testo mediante la semplice analisi delle frequenze delle singole lettere consiste nel cifrare insieme coppie di lettere e un semplice schema per farlo è il cifrario di Playfair. Si sceglie una parola chiave e vi si eliminano le lettere duplicate, poi la si inserisce, seguita ordinatamente dalle altre lettere dell'alfabeto inglese (non presenti dunque nella parola chiave), in una scacchiera 5 × 5 (dato che ci sono soltanto 25 caselle e l'alfabeto inglese ha 26 lettere, la I e la J sono considerate indistinguibili. Verificare che nel testo sorgente le lettere siano pari, sennò aggiungere una "Z").
Ecco lo schema che si ottiene usando PLAYFAIR come parola chiave:

P	L	A	Y	F
I	R	B	C	D
E	G	H	K	M
N	O	Q	S	T
U	V	W	X	Z

Continua nella prossima slide…

Esercizio 7

Per cifrare una coppia di lettere, ad esempio AM, si cerca il rettangolo che ha A e M negli angoli.

P	L	A	Y	F
I	R	B	C	D
E	G	H	K	M
N	O	Q	S	T
U	V	W	X	Z

La codifica di questa coppia si ottiene prendendo gli altri due angoli del rettangolo, in questo caso FH. Se le due lettere della coppia si trovano sulla stessa riga o sulla stessa colonna, come GO, vengono semplicemente scambiate tra loro. La decifrazione avviene nello stesso modo. Scrivete un programma che cifri o decifri un file di testo secondo lo schema qui presentato. [P7.23]

Esercizio 7 – *il codice python*

```python
def main():
    # Gather input from the user.
    keyword = removeDuplicates(input("Enter the keyword: ").upper())
    in_name = input("Enter the input file name: ")
    out_name = input("Enter the output file name: ")

    # Read all of the text out of the file.
    inf = open(in_name, "r")
    text = inf.read()
    inf.close()

    # I's and J's are considered the same, so replace all J's with I's.
    text = text.replace("J", "I")

    # Ensure we have an even number of letters by padding with a Z.
    count = 0
    for ch in text:
        if ch in "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz":
            count = count + 1

    if count % 2 == 1:
        text = text + "Z"

    # I's and J's are considered the same, so replace all J's with I's.
    keyword = keyword.replace("J", "I")

    # Create the table for modifying the text.
    table = createTable(keyword)
```

Esercizio 7 – *il codice python*

```
# Perform the transformation.
result = ""
i = 0
while i < len(text):
    before = ""
    between = ""
    after = ""

    while i < len(text) and text[i] not in "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        before = before + text[i]
        i = i + 1

    (r1, c1) = getPos(text[i], table)
    i = i + 1

    while i < len(text) and text[i] not in "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        between = between + text[i]
        i = i + 1

    (r2, c2) = getPos(text[i], table)
    i = i + 1

    while i < len(text) and text[i] not in "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz":
        after = after + text[i]
        i = i + 1

    # Handle two letters in the same row.
    if r1 == r2:
        result = result + before + table[r1][c2] + between + table[r2][c1] + after
    # Handle two letters in the same column.
    elif c1 == c2:
        result = result + before + table[r2][c2] + between + table[r1][c1] + after
    # Handle all other cases.
    else:
        result = result + before + table[r1][c2] + between + table[r2][c1] + after

# Save the result.
outf = open(out_name, "w")
outf.write(result)
outf.close()
```