Ancora sui file

Errori di apertura

- L’operazione di apertura del file è fondamentale, sia in caso di lettura, sia in caso di scrittura.
- In entrambi i casi è necessario assicurarsi che il file sia stato aperto in maniera corretta
  - in lettura è ovviamente anche necessario verificare che il file specificato sia preesistente
- Per verificare il successo o meno di un’operazione di apertura di un file, si controlla se l’oggetto associato al file sia stato istanziato o meno

Gestione errori di apertura: esempio (1)

```
…
mioFile.open("dati.txt");
if(mioFile){
        … //apertura OK, uso il file
} else{
        cout<<"Errore"; // errore
}
```

- Dopo l'apertura del file
  - se l'oggetto mioFile è istanziato (condizione dell'if valutata a true) allora il file può essere usato
  - Altrimenti si stampa un messaggio di errore

is_open() e fail()

- Per assicurarsi che il file sia stato aperto in maniera corretta è anche possibile usare i metodi ***is_open()*** o ***fail(),*** che restituiscono entrambi un valore booleano:
  - *is_open()* restituisce **true** se il file è aperto correttamente, **false** altrimenti;
  - *fail()* ritorna **true** se il file non è aperto correttamente, **false** altrimenti.

Gestione errori di apertura: esempio (2)

```
…
mioFile.open("dati.txt");
if(mioFile.is_open()){
        … //apertura OK, uso il file
} else{
        cout<<"Errore«; // errore
}
```

```
…
mioFile.open("dati.txt");
if(!mioFile.fail()){
        … //apertura OK, uso il file
} else{
        cout<<"Errore«; // errore
}
```

Lettura con getline()

- Il linguaggio C++ consente di leggere un'intera linea di testo da un file utilizzando il metodo *getline()*

  getline(variabile_ifstream, variabileStringa)

- Ovviamente per leggere tutte le righe di un file si utilizza un ciclo di lettura che le legge tutte fino a quando viene raggiunta la fine del file.

Lettura di file con getline()

```
string riga;
…
//fino a quando ci sono righe nel file
while (getline(miofile, riga)) {
        … //uso i dati della riga letta
}
miofile.close();
```

- La condizione del ciclo while controlla se la riga letta con getline() è valida:
  - *getline(*miofile*, riga)* viene valutata a *true* se viene letta una riga valida: in tal caso viene eseguito il corpo del ciclo;
  - quando si raggiunge la fine del file, *getline(*miofile*, riga)* restituisce *null*, che corrisponde a *false* e quindi il ciclo termina.

File con caratteri delimitatori

- In alcuni casi le informazioni presenti su una stessa riga di un file vengono distinte grazie a un carattere separatore
  - il punto e virgola e il carattere ASCII per la tabulazione sono molto diffusi
- Un esempio tipico in questo senso è dato dai file in formato **CSV** (***Comma Separated Values***), che sono spesso utilizzati per l'importazione e l'esportazione di una tabella di dati da applicazioni come i gestori di fogli di calcolo o di database.

ESEMPIO DI FILE CSV

Intestazione →

Contatti →

Cognome	Nome	Indirizzo	CAP	Città
Rossi	Marco	Via Roma, 12	57100	Livorno
Neri	Andrea	Via del Mare, 15	56100	Pisa
Bianchi	Giovanni	Via Ferruccio, 7	55100	Firenze

Cognome;Nome;Indirizzo;CAP;Città<CR><LF>
Rossi;Marco;Via Roma, 12;57100;Livorno<CR><LF>
Neri;Andrea;Via del Mare, 15;56100;Pisa<CR><LF>
Bianchi;Giovanni;Via Ferruccio, 7;55100;Firenze<CR><LF>

- La tabella presentata, se esportata in un file in formato CSV, potrebbe assumere il seguente formato, dove il simbolo «;» è il carattere separatore utilizzato per distinguere i singoli valori.

Lettura con getline() in file con delimitatori

- Nel caso di informazioni separata da un carattere delimitatore si può usare per la lettura una variante della funzione getline()

  getline(var_ifstream, var_Stringa, delimitatore)

- In questo caso la lettura si ferma al carattere delimitatore (che viene scartato).

Lettura di file con getline()

Cognome;Nome;Indirizzo;CAP;Città<CR><LF>
Rossi;Marco;Via Roma, 12;57100;Livorno<CR><LF>
Neri;Andrea;Via del Mare, 15;56100;Pisa<CR><LF>
Bianchi;Giovanni;Via Ferruccio, 7;55100;Firenze<CR><LF>

```
…
fc.open('contatti.txt');
string cogn, nome, ind, CAP, citta;
…
getline(fc, cogn, ';');
getline(fc, nome, ';');
getline(fc, ind, ';');
getline(fc, CAP, ';');
getline(fc, citta);
…
```

- Consideriamo il file contatti.txt qui a sinistra.
- Per leggere i dati di una riga separati dal ';' usiamo la funzione getline indicando come carattere delimitatore il punto e virgola ';'.

ESERCIZI

1. Scrivere un programma C++ che copi un file di testo origine.txt in uno destinazione.txt, trasformando tutte le lettere in maiuscolo.
2. Modificare il programma precedente richiedendo il nome del file da tastiera all’utente.

ESERCIZIO

- Scrivere un programma C++ che gestisca i dati di alcuni studenti (max 50). In particolare, i dati sono contenuti in un file di testo studenti.txt, dove ogni riga contiene il nome, il cognome e la media dei voti di un singolo studente. I dati su una singola riga sono separati dal «;».

  Valentina;Monreale;7
  Antonio;Benedetto;8
  ….

- Il programma leggerà i dati dal file che saranno caricati in un vettore. Ogni elemento del vettore sarà di tipo studente, cioè una struttura composta da nome, cognome e voto.
- Il programma ricercherà e stamperà il massimo voto nel vettore e calcolerà il voto medio, mediante opportune funzioni.

ESERCIZIO 3

- Scrivere un programma C++ che gestisca i dati di alcuni prodotti (max 50). In particolare, i dati sono contenuti in un file di testo prodotti.txt, dove ogni riga contiene il nome, la quantità e il prezzo di un singolo prodotto. I dati su una singola riga sono separati dal «;».

  tavolo;7;50.50

  sedie legno;8;15.75

  ….

- Il programma leggerà i dati dal file che saranno caricati in un vettore. Ogni elemento del vettore sarà di tipo prodotto.
- Il programma ricercherà e stamperà il prezzo minimo nel vettore e stamperà i prodotti con quantità zero, mediante opportune funzioni.