GESTIONE DEGLI STREAM

Libreria Standard di I/O in C++

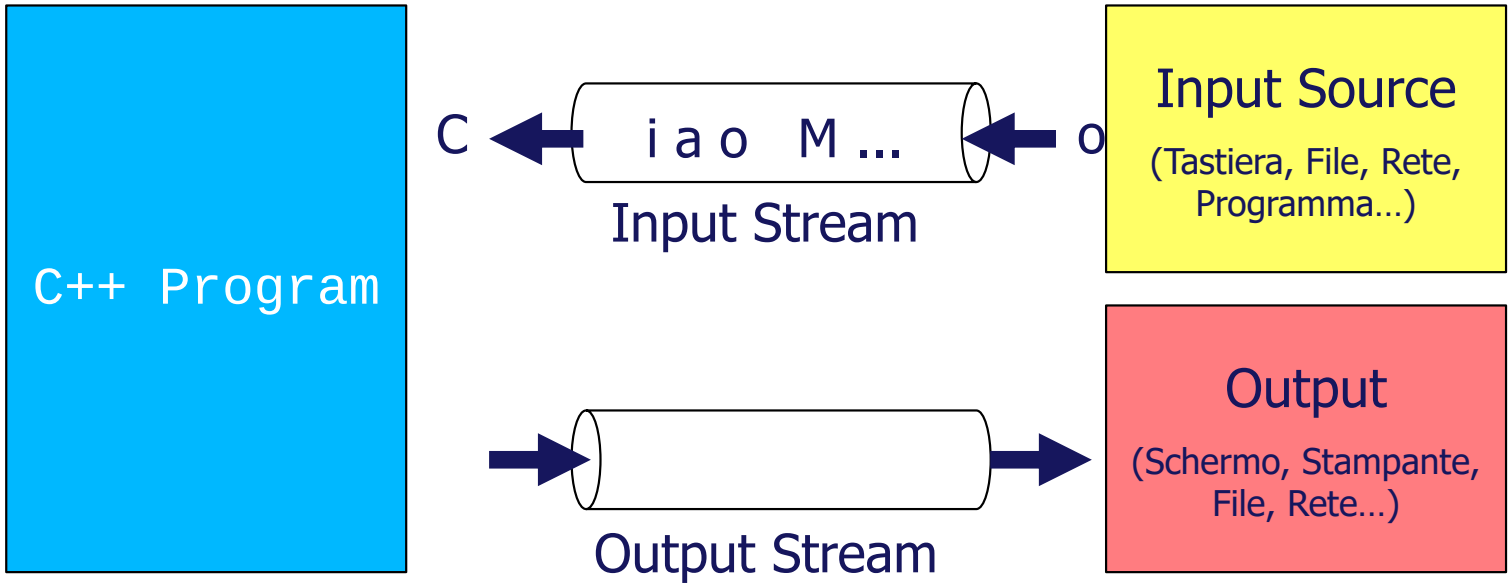
Luigi Coppolino, Luigi Romano

ROADMAP

- Il concetto di stream e la gerarchia delle librerie standard
- Stream I/O
  - Formattazione stringhe
- Gestione dei File
- Esempi
- Esercizi proposti

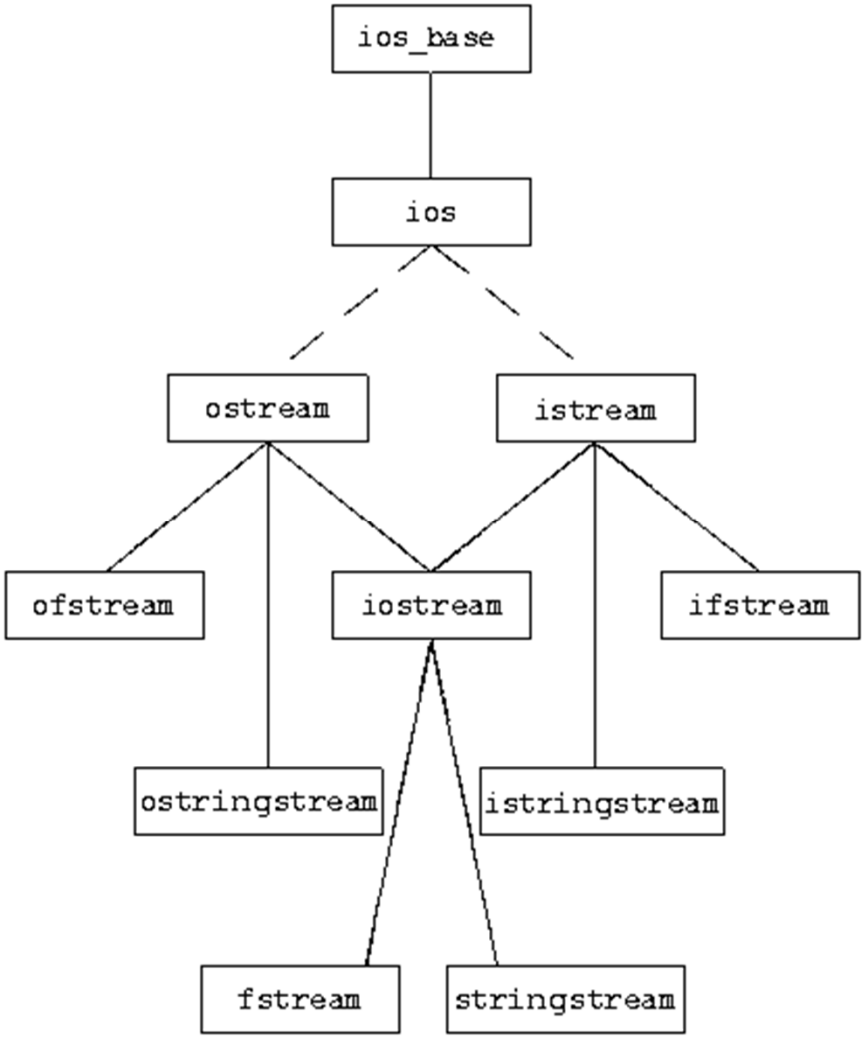
ASTRAZIONE DELLO STREAM

- No builtin I/O
- Uso di una libreria STL (<iostream>, <stdio>)
- Astrazione di STREAM



- Binary vs Character streams

GERARCHIA DI CLASSI I/O (SEMPLIFICATA)



INPUT STREAM

- istream: gestisce lo standard input attraverso l'oggetto *cin*

```
class istream: virtual public ios  // iostream
{…
};
```

- L'operatore >> (estrattore) accetta tutti i tipi fondamentali del C++

```
oggetto_istream >> oggetto_variabile
```

- Estrae testo da `oggetto_istream` e lo trasferisce in `oggetto_variabile`

- Es. cin >> eta;

STATO DI ISTREAM

- Quattro variabili boolean (da classe base ios)

`good` : lo stato è buono

`bad` : qualcosa di scorretto nel flusso

`fail` : ultima operazione di estrazione ha dato errore

`eof` : carattere EOF riscontrato

- Es.

```
int a;
cin >> a; //input 5a4 => bad=true;fail=true;good=false..
```

- I singoli valori di stato possono essere recuperati con metodi pubblici aventi lo stesso nome
  - Es. `cin.good(); // true se tutto ok`
- `.rdstate()` legge lo stato globale
- `.clear()` ripristina lo stato

LIMITI DELL'ESTRATTORE >>

- Usando l'estrattore, ad esempio con tipo intero, il newline resta nello stream

```
int a;
cin >> a; // con input 123↵
```

- Dallo stream vengono prelevati i caratteri 1 2 e 3 convertiti in intero mentre EOL resta nello stream

```
int a;
string s;
cin >> a;
cin >> s;
```

  - a=123
  - s=\n

- Se cin >> s riceve un input con spazi l'estrattore si arresta allo spazio "paolino paperino" s="paolino"

METODI CLASSE ISTREAM

- get()/get(char& c) legge un singolo carattere dallo stream

```
while (cin.get(c))  // c=cin.get()
    std::cout << c;
```

- istream& getline (char* s, streamsize n);
- istream& getline (char* s, streamsize n, char delim);

```
char name[256];
std::cout << "Please, enter your name: ";
std::cin.getline (name,256) // “Paolo Paperino”
```

- Non si interrompe allo spazio e inserisce "\n"
- istream& ignore (streamsize n = 1, int delim = EOF);
  - Estrae e scarta caratteri dallo stream `cin.ignore()`

OUTPUT STREAM

- ostream: gestisce lo standard output
  - cout : invia a standard output -> Schermo
  - cerr : invia a standard error -> Schermo
  - clog: come per cerr ma prima bufferizza lo stream in memoria
- ostream: gestisce lo standard input attraverso l'oggetto *cout*

```
class ostream: virtual public ios  // iostream
{…
};
```

- L'operatore << (insertore) accetta tutti i tipi fondamentali del C++

```
oggetto_ostream << oggetto_variabile
```

- Estrae testo da `oggetto_variabile` e lo trasferisce in `oggetto_ostream`
- Es. cout << eta;
- Derivando da `ios` ha gli stessi attributi di stato di `istream`

METODI CLASSE OSTREAM

- streamsize width (streamsize wide); // setta la larghezza del campo out
- streamsize width() const; // ritorna larghezza del campo out
- char fill (char fillch); // carattere di riempimento
- char fill() const; // ritorna il carattere di riempimento

```
cout << 100 << '\n'; // 100
cout.width(10);
cout << 100 << '\n'; //         100
cout.width(10);
cout.fill('x');
cout << 100 << '\n'; // xxxxxxx100
```

- streamsize precision(streamsize prec); // setta la precision (# decimali)
- streamsize precision() const; // ritorna la precisione

FORMATTAZIONE DI STRINGA (EREDITA DA IO.H)

Numerical base format flags ("basefield" flags):

dec	Use decimal base (function)
hex	Use hexadecimal base (function)
oct	Use octal base (function)

Floating-point format flags ("floatfield" flags):

fixed	Use fixed floating-point notation (function)
scientific	Use scientific floating-point notation (function)

Adustment format flags ("adjustfield" flags):

internal	Adjust field by inserting characters at an internal position (function)
left	Adjust output to the left (function)
right	Adjust output to the right (function)

```
cout.width(10);
cout << right << dec << n << '\n'; //       70
cout.width(10);
cout << right << hex << n << '\n';  //       46
cout.width(10);
cout << right << oct << n << '\n';  //      106
```

FORMATTAZIONE DI STRINGA <IOMANIP>

Parametric manipulators

setiosflags	Set format flags (function)
resetiosflags	Reset format flags (function)
setbase	Set basefield flag (function)
setfill	Set fill character (function)
setprecision	Set decimal precision (function)
setw	Set field width (function)
get_money C++11	Get monetary value (function)
put_money C++11	Put monetary value (function)
get_time C++11	Get date and time (function)
put_time C++11	Put date and time (function)

http://www.cplusplus.com/reference/iomanip/

GESTIONE DEI FILE

- La gestione dei file avviene mediante le classi
  - ifstream
  - ofstream
  - fstream (che include iostream)

SCRITTURA SU FILE

- Per scrivere su file va aperto un flusso in output
  - `ofstream fout("nomefile", mode);`
- mode può essere:
  - `ios::out` (se il file esiste viene sovrascritto) (default)
  - `ios::append` (scrittura in coda a quanto già esistente)
- Es. salvare su file l'input dell'utente

```
ofstream fout(nome_file);
if(!fout){
   cout << "Errore nell'apertura del file";
   return -1;
}
while(cin.get(c)) fout.put(c); // per terminare
                               // usare Ctrl+D che
                               // equivale al fine file
```

LETTURA DA FILE

- Per leggere da un file, bisogna aprire un flusso in input da file
  - `ifstream fin("nomefile");`
  - fin assume valore nullo se l'apertura del file non ha successo (es. file inesistente)

A questo punto fin è un normale stream da cui poter leggere caratteri ad esempio con `fin.get(c)` (quando raggiunge fine file ritorna `false`)

- Es. mostrare a video il contenuto di un file:

```
ifstream fin(nome_file);
if(!fin){
   cout << "Errore nell'apertura del file";
   return -1;
}
char c;
while(fin.get(c)) cout << c;// termina con fine file
```

APERTURA IN INPUT/OUTPUT

- La classe `fstream` consente di aprire un file in input/output
- Una volta creato l'oggetto si può usare il metodo `open()`
  - `void open (const char* filename, int mode = ios::in | ios::out);`
- mode può essere

member constant	stands for	access
in	**in**put	File open for reading: the *internal stream buffer* supports input operations.
out	**out**put	File open for writing: the *internal stream buffer* supports output operations.
binary	**binary**	Operations are performed in binary mode rather than text.
ate	**at e**nd	The *output position* starts at the end of the file.
app	**app**end	All output operations happen at the end of the file, appending to its existing contents.
trunc	**trunc**ate	Any contents that existed in the file before it is open are discarded.

ESEMPI

```
fstream f;
char nomefile[] = "temp.txt";
f.open(nomefile, ios::in); // file in input

f.open(nomefile, ios::out); // output

f.open(nomefile, ios::out | ios::in |
          ios::binary); //input+output+binario

f.close() // Chiude il file
```

ULTERIORI NOTE

- `open()` non ha valore di ritorno, per cui in caso di errore questo non viene notificato
- Meglio usare le versioni con costruttore `ifstream if(nomeFile);` su cui è possibile fare controlli
- Se lo stream è aperto in modo binario, anziché usare insertore (<<) ed estrattore (>>), bisogna usare i metodi `put()` `get()` `write()` e `read()` ereditati da `iostream`

UN PROGRAMMA CHE SALVA SU FILE UNA RUBRICA


```
// nome file scriviRubrica.cpp
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

struct Persona{
     string nome;
     string cell;
     int  eta;
};

int main(int argc, char* arg[]){//nomeprogramma parametro1 parametro 2…
     if (argc != 2){
          cout << "Errore, uso corretto: scriviRubrica nome_file" << endl;
          return -1;
     }
     const int N = 3;
```

UN PROGRAMMA CHE SALVA SU FILE UNA RUBRICA


```
// nome file scriviRubrica.cpp
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

struct Persona{
     string nome;
     string cell;
     int  eta;
};

int main(int argc, char* arg[]){
     if (argc != 2){
          cout << "Errore, uso corretto: scriviRubrica nome_file" << endl;
          return -1;
     }
```

Un programma di nome xxx può essere invocato con parametri:

xxx param1 param2 …

argc = 3

argv[0]="xxx"

argv[1]="param1"

argv[2]="param2"

Es.

g++ scriviRubrica.cpp –o prg

argc=4

argv[0]="g++"

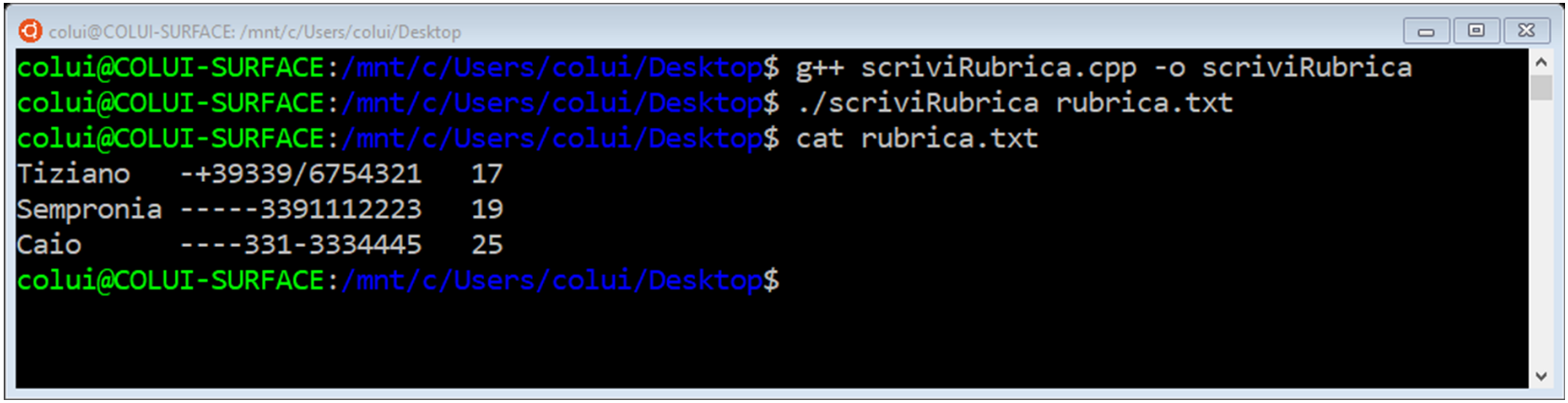
argv[1]="scriviRubrica.cpp"

argv[2]="-o"

argv[3]="prg"

```
    const int N = 3;
    Persona p[N];
    p[0].nome="Tiziano";
    p[0].cell="+39339/6754321";
    p[0].eta=17;
    p[1].nome="Sempronia";
    p[1].cell="3391112223";
    p[1].eta=19;
    p[2].nome="Caio";
    p[2].cell="331-3334445";
    p[2].eta=25;
    ofstream fo(arg[1],ios_base::app);
    if (!fo) {
        cout << "Impossibile aprire il file" << arg[1];
        return -1;
    }
    for(int i=0; i < N; i++){
        fo   << setw(10) << setfill(' ') << left << p[i].nome << setw(15)
             << setfill('-') << right << p[i].cell << setw(5) << setfill(' ‘)
             << right << p[i].eta << endl;
    }
}
```

ESECUZIONE

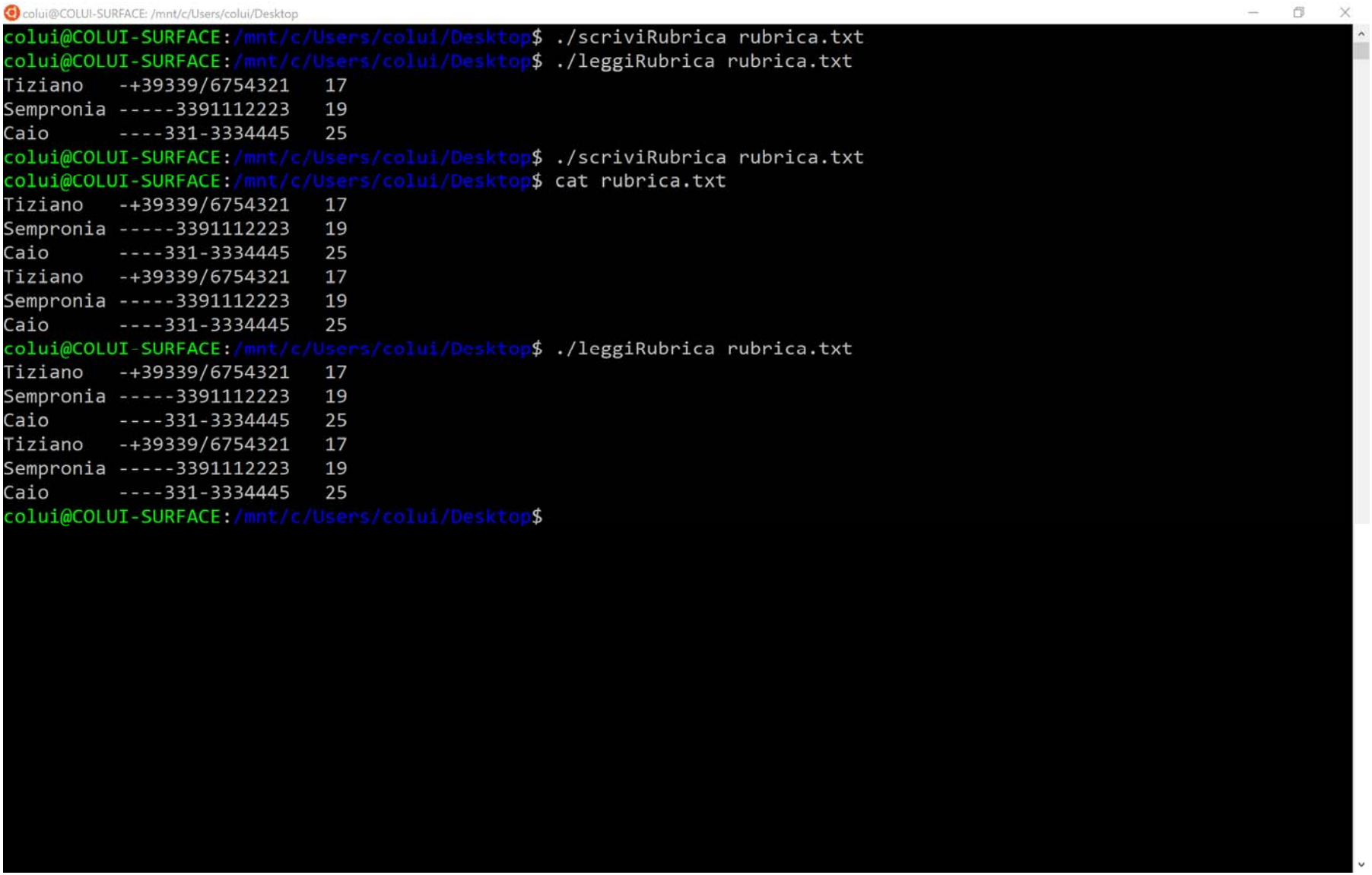

colui@COLUI-SURFACE: /mnt/c/Users/colui/Desktop
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ g++ scriviRubrica.cpp -o scriviRubrica
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ ./scriviRubrica rubrica.txt
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ cat rubrica.txt
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$

UN PROGRAMMA CHE LEGGE DA FILE UNA RUBRICA


```
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

struct Persona{
    string nome;
    string cell;
    int eta;
};
int main(int argc, char* arg[]){
    if (argc != 2){
        cout << "Errore, uso corretto: leggiRubrica nome_file"
                << endl;
        return -1;
    }
```

```
    Persona p;

    ifstream fi(arg[1]);
    if (!fi ) {
        cout << "Impossibile aprire il file" << arg[1];
        return -1;
    }
    while (true){
        fi >> p.nome;
        fi >> p.cell;
        fi >> p.eta;
        if ( fi.eof() ) return 1;
        cout    << setw(10) << setfill(' ') << left << p.nome
                << setw(15) << setfill('-') << right << p.cell
                << setw(5) << setfill(' ') << right << p.eta << endl;
    }
}
```

ESECUZIONE


colui@COLUI-SURFACE: /mnt/c/Users/colui/Desktop
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ ./scriviRubrica rubrica.txt
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ ./leggiRubrica rubrica.txt
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ ./scriviRubrica rubrica.txt
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ cat rubrica.txt
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$ ./leggiRubrica rubrica.txt
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
Tiziano -+39339/6754321 17
Sempronia -----3391112223 19
Caio ----331-3334445 25
colui@COLUI-SURFACE:/mnt/c/Users/colui/Desktop$

ESERCIZI DI AUTOVALUTAZIONE

- Creare una classe Rubrica che può contenere fino a 100 persone, consente di inserire una nuova persona, stampare l'elenco, salvare su file, caricare da file.
- Aggiungere alla classe Rubrica la possibilità di ordinare l'elenco di Persone e cercare una persona.