UNIVERSITÀ
degli STUDI
di CATANIA

Gestione errori di IO in C++

Corso di programmazione I AA 2019/20

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella
Web: http://www.dmi.unict.it/farinella
Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

Metodi `fail()`, `ignore()` e `clear()`

```
1  float x; //floating point to collect the user input
2  std::cout << "Insert any number: " << endl;
3  std::cin >> x;
```

Se l'utente inserisce una sequenza di caratteri che non rappresenta un numero (Ad esempio: "pippo")?

Nella variabile `x` non sarà copiato alcun valore! (errore di IO)

L'occorrenza di un qualsiasi errore di IO si potrà verificare tramite il metodo `std:ios_base.fail()` oppure mediante l'operatore '!' usato sull'oggetto che rappresenta lo stream (in questo caso `cin`).

Metodi `fail()`, `ignore()` e `clear()`

1) metodo `std::basic_ios.fail()`

```
1  float x;
2  std::cin >> x;
3  if(cin.fail()){ //IO error
4     cerr << "Inserito input non valido!" << endl;
5     return -1;
6  }
```

Metodi `fail()`, `ignore()` e `clear()`

2) operatore '!' della classe `std:basic_ios`.

```
1  float x;
2  if(!(std::cin >> x)){ //IO error
3    std::cerr << "Inserito input non valido!" << endl;
4    return -1;
5  }
```

`!(std::cin >> x)` restituisce un bool.

Metodi `fail()`, `ignore()` e `clear()`

Quando il metodo `std::basic_ios.fail()` restituisce `true`?

La classe `std::ios_base` contiene alcuni "flags" che rappresentano lo *stato* dello stream.

- `eofbit` (End Of File)
- `failbit` (Errore di IO: formattazione o estrazione)
- `badbit` (Altri errori)
- `goodbit` (nessun errore)

Metodi `fail()`, `ignore()` e `clear()`

Il risultato della invocazione del metodo `fail()` dipenderà da una opportuna dei tre flag.

Per approfondire:
https://en.cppreference.com/w/cpp/io/ios_base/iostate

Metodi `fail()`, `ignore()` e `clear()`

ios_base::iostate flags			basic_ios accessors					
eofbit	failbit	badbit	good()	fail()	bad()	eof()	operator bool	operator!
false	false	false	true	false	false	false	true	false
false	false	true	false	true	true	false	false	true
false	true	false	false	true	false	false	false	true
false	true	true	false	true	true	false	false	true
true	false	false	false	false	false	true	true	false
true	false	true	false	true	true	true	false	true
true	true	false	false	true	false	true	false	true
true	true	true	false	true	true	true	false	true

Fonte:

https://en.cppreference.com/w/cpp/io/ios_base/iostate

`ios_base.fail()` true $\iff$ (`failbit==true` || `badbit==true`).

EOF va controllato a parte con metodo `eof()`.

Metodi `fail()`, `ignore()` e `clear()`

Esempi

`A13_01_FAIL.cpp`

Metodi `fail()`, `ignore()` e `clear()`

Codice più "robusto".

```
1  #include <iostream>
2  #include <limits>
3  float x;
4  cout << "Insert any number: " << endl;
5  cin >> x;
6  if(cin.fail()){ //IO error!
7    cerr << "IO error: cin.fail()=" << cin.fail() << endl;
8    cin.clear();  //RESET Io flags
9    cout << "cin.fail()=" << cin.fail() << endl;
10 }
11 else
12   cout << "The number you entered is " << x << endl;
13
14 cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

Metodi `fail()`, `ignore()` e `clear()`

```
1  if(cin.fail()){ //IO error!
2    //...
3    cin.clear();  //RESET IO flags
4    //...
5  }
```

`cin.clear()` **pone a zero tutti i flag di errore dello stream.**

Non si potrà procedere con altre operazioni di IO **senza aver prima posto a zero i flags di errore.**

Metodi `fail()`, `ignore()` e `clear()`

```
cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

prototipo funzione:

```
ignore(std::streamsize count=1, int_type delim=Traits::eof());
```

il metodo `std::basic_istream::ignore` permette di scartare caratteri rimasti nello stream:

- a seguito di **errore**, ad esempio utente ha inserito "pippo" anziche un numero
- perchè l'utente ha inserito **più stringhe** separate da spazi.

Metodi `fail()`, `ignore()` e `clear()`

```
cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

```
ignore(std::streamsize count=1, int_type delim=Traits::eof());
```

- `count` rappresenta il **numero di caratteri** da scartare dal buffer dello stream, se specificato `numeric_limits<streamsize>::max()` allora i caratteri non vengono conteggiati ma si considera solo il delimitatore;
- delim rappresenta un carattere di "delimitazione" oltre il quale non saranno scartati ulteriori caratteri.

Metodi `fail()`, `ignore()` e `clear()`

```
1    bool eof=false;
2    if(eof=cin.eof()){ //End Of File!
3       cout << "EOF!.. cin.fail()=" << cin.fail() << endl
4       cin.clear(); //RESET Io flags
5       //...
6    }
```

Utente può digitare il carattere di `EOF` tramite tastiera con la combinazione di tasti `CTRL`+d.

Metodi `fail()`, `ignore()` e `clear()`

Esempi

A13_01_multiple_input.cpp

A13_02_EOF_multiple_input.cpp

A13_03_LOOP.cpp

A13_04_LOOP_multiple_input.cpp

FINE