Libreria iostream e funzioni di I/O

Anna Corazza

aa 2023/24

Dove studiare

- Str'13, sezione 4.3, 38
- Str'14, capitolo 10

Str'13 Bjarne Stroustrup, The C++ Programming Language (4th edition), 2013 `https://www.stroustrup.com/4th.html`

Str'14 Bjarne Stroustrup, Programming: Principles and Practice using C++ (2nd edition), 2014 `https://www.stroustrup.com/programming.html`

Stream I/O

- Provvisto dalla libreria standard `iostream`: input e output di carattere formattato.
- Le operazioni di input sono tipate e estendibili, in modo da poter gestire tii definiti dall'utente.
- Altri tipi di interazione con l'utente (grafico, ad esempio) non sono inclusi nello standard ISO e quindi nemmeno in questa libreria.

Output

- Un `ostream` converte un oggetto con tipo in uno stream di caratteri (bytes).
- Si può costruire l'output per ogni tipo definito dall'utente.
- Operatore « ("put to") definito per tutti gli oggetti di tipo `ostream` (es: lo standard output `cout`, lo standard error `cerr`, non bufferizzato o `clog`, bufferizzato).
- Per default, i valori scritti su uno stream di uscita sono convertiti in una **sequenza di caratteri**
- esempio: l'intero 10 sarà convertito nella sequenza '1','0'.
- Ogni operazione di « restituisce lo stream, di modo da poter concatenare diverse operazini:

```
int i=23;
cout << "Risultato " << i << '\n';
```

- L'output di un carattere (ad esempio, 'a') risulta nel carattere, e non nella sua codifica.

Input

- Un `istream` converte uno stream di caratteri in oggetti con tipo.
- `istream` per input di tipi built-in, estendibile a tipi definiti dall'utente.
- Operatore » ("get from") definito sugli oggetti di tipo `istream`, tra cui `cin`.
- Il tipo dell'operando a destra di » determina:
  - quale ingresso accetta;
  - l'obiettivo dell'operazione.

```
// legge un intero e lo mette in i
int i;
cin >> i;

// legge un double e lo mette in d
double d;
cin >> d;
```

Input – continua

- Per leggere una sequenza di caratteri useremo `string`; tuttavia la lettura si ferma al pimo carattere non alfanumerico:

```
// legge un intero e lo mette in i
string str; // std::string
cin >> str;
```

- Per leggere un'intera linea dobbiamo usare `getline()` (vedi esempio)
- Per leggere un solo carattere `get()`

I/O per tipi definiti dall'utente

«

- La libreria `iostream` permette di definire operazioni di I/O per i tipi definiti dall'utente.

```
struct Cliente{
  string nome;
  int numero;
};

ostream& operator<<(ostream& os, const
    Cliente& e){
  return os << "{\"" << e.nome << "\"," <<
      e.numero << "}";
}
```

- Nella definizione dell'operatore « per il nuovo tipo, lo stream di uscita
  - viene preso, per riferimento, come primo argomento
  - viene restituito come risultato

Altro

- Input per tipi definiti dall'utente: Si veda l'esempio 18.
- Gli stream possono essere associati a file (`fstream`) o a stringhe (`sstream`).

Gestione degli errori

- Un `iostream` si può trovare in uno tra quattro stati:
  - `good()` : la precedente operazione di `iostream` ha avuto successo
  - `eof()` : arrivati alla fine dell'ingresso ("end-of-file")
  - `fail()` : qualcosa di inaspettato (per esempio, mi ha aspetta una cifra e ho trovato un carattere
  - `bad()` : qualcosa di seriamente inaspettato (ad esempio, un errore nella lettura del disco)
- Qualsiasi operazione venga tentata su di uno stream che non termina con uno stato `good()` non ha nessun effetto.
- Un `iostream` può venire usato come condizione: `true` solo se si trova nello stato `good()`

Ripristino dello stream

- Dopo un errore di lettura, per pulire lo stream e procedere:

```
int i;
if(cin>>i){
  // fa quello che devi
} else if(cin.fail()){
  cin.clear();
  string s;
  if(cin>>s){ // vediamo cosa contiene
    // procedi
  }
}
```

- In alternativa, possiamo usare le eccezioni . . .