

Le librerie `std::string` e `random`

Anna Corazza

aa 2023/24

Dove studiare

- string** ▶ Str'13, sezioni 4.2, 36
 ▶ Str'14, sezioni 23.2, 27.5, B.8
- random** ▶ Str'13, sezioni 5.6.3, 40.7
 ▶ Str'14, sezioni 24.7, B.9.6

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>

La libreria standard <string>

- ▶ Complementa quello che abbia visto sui letterali stringa
- ▶ Concatenazione (+)

```
string compose(const string& name, const
               string& domain){
    return name+'@'+domain;
}
auto
    addr=compose("anna.corazza","unina.it");
```

- ▶ Operatore (+=)

```
string addNewline(string& s1, string& s2){
    s1 = s1 + '\n';
    s1 += '\n';
}
```

- ▶ In C non esiste un tipo dedicato per le stringhe, ma sono un insieme di convenzioni supportate da alcune funzioni.

string

- ▶ Una stringa è mutable.
- ▶ Operazioni di indicizzazione () e di sottostringa
- ▶ Manipolazione di sottostringhe:

```
string name = "Niels Stroustrup";  
void func() {  
    string s = name.substr(6,10); //  
        "Stroustrup"  
    name.replace(0,5,"nicholas");  
    name[0]=toupper(name[0])  
}
```

- ▶ `sustr()` restituisce una `string` che è una copia della stringa indicata dagli argomenti (inizio e lunghezza).
- ▶ `replace()` sostituisce una sottostringa con una stringa, anche di lunghezza diversa.
- ▶ Nei confronti tra stringhe o con letterali stringa si considera l'ordine lessicografico.

string

Costruttori

```
string s0; // la stringa vuota
string s1("Inizializzazione semplice");
string s2(s1); // copia una stringa
string s3(7, 'a') // "aaaaaaa"
string s4(0) // in C ok, in C++ Pericoloso!
```

► Usando i puntatori:

```
string s5(nullptr); // pericolosissimo!
string s6(p); // dipende dal valore di p
string s7{"OK"}; // OK puntatore a una
                stringa in stile C
```

string

Metodi

- ▶ `s.size()`: numero di caratteri in `s`
- ▶ `s.length()`: lo stesso
- ▶ `s.clear()`
- ▶ `s.empty()`: **bool**, è vuota?
- ▶ `s.front()`: `s[0]`
- ▶ `s.back()`: `s[s.size()-1]`

Numeri pseudo-random

Libreria std <random>

► Due parti

1. un **algoritmo** che genera numeri (pseudo-)casuali secondo una distribuzione uniforme;
2. una **distribuzione** che mappa questa distribuzione in una distribuzione matematica in un certo intervallo (es:

`uniform_int_distribution,`
`normal_distribution,`
`exponential_distribution, ...)`

```
// uso un alias per leggibilita '  
using my_engine=default_random_engine;  
using my_distribution= uniform_int_distribution<int>;
```

```
my_engine re {};  
my_distribution one_to_six {1,6};  
auto dado = bind(one_to_six,re);
```

```
// per usare il generatore dato  
int x=dado();
```

► `seed()`