Standard library `stdlib`

Palestra di algoritmi

2020-2021

La standard library

stdlib

- Contiene le principali funzioni/oggetti utilizzati solitamente:
  - cin, cout
  - ifstream, ofstream
  - sort(...), abs(...), sqrt(...), ...
- È una collezione di librerie:
  - iostream, fstream, cmath, algorithm, ...
  - Si possono includere tutte con

```
#include <bits/stdc++.h>
```

La standard library

Altre strutture utili

- vector
- string
- pair
- queue
- stack
- set
- map

Vector

Simile ad un array, solitamente più comodo.

Dichiarazione

```
vector<long>  v1; // vector vuoto
vector<int> v2(10, 0); // 10 elementi tutti 0
vector<MyStruct> v3 = {x,y,z}; // vector di 3 elementi
```

Metodi utili

v1[i]=1	accede all'i-esimo elemento di v1
v1.size()	restituisce il nr. di elementi di v1
v1.push_back(x)	aggiunge x alla fine
v1.pop_back()	rimuove l'ultimo elemento
v1.resize(n)	ridimensiona il vettore

Vector

Esempio

```
int n, e;
vector<int> v;
cin >> n;
for(int i = 0; i < n; i++){
    cin >> e;
    v.push_back(e);
}
// ordinamento
sort(v.begin(), v.end());
// stampa
for(int item : v)
    cout << item << endl;
```

String

Simile ad un char[N], molto più comodo.

Dichiarazione

```
string s1; // string vuota
string s2(10, 'a'); // stringa di 10 'a'
string s3 = "helloo!"
```

Metodi utili

s1[i]='b'	modifica l'i-esimo carattere
s1.push_back(x)	aggiunge il carattere x alla fine
s1.pop_back()	rimuove l'ultimo carattere
s1.resize(n)	ridimensiona la stringa
s1.substr(pos, len)	restituisce la sottostringa s1[pos:pos+len]
s1 == s2	true sse s1, s2 sono uguali
s1 ¡ s2, s1 $\leq$ s2	ordine lessicografico

String

Esempio

```
string s1, s2;
cin >> s1;
s2 = s1.substr(1, s1.size()-2);
cout << s2;
```

Pair

Coppia di valori (di tipi diversi). Si usa quando non c'è bisogno di definire una struct ad-hoc.

Dichiarazione

```
pair<string, vector<int>> p1;
pair<int, long> p2 = {x, y};
```

Metodi utili

`p1.first` accede al primo elemento

`p1.second` accede al secondo elemento

Pair

Esempio

```
/**
 * Date coppie nome cognome stamparle in ordine
 * alfabetico di cognome, nome
 */
vector<pair<string, string>> v;
for(int i = 0; i < n; i++){
    pair<string, string> tmp;
    // leggo il pair <cognome, nome>
    cin >> tmp.second >> tmp.first;
}
// a < b sse a.first < b.first OR
// a.first == b.first AND a.second <= b.second
sort(v.begin(), v.end());
for(auto &p : v)
    cout << p.second << " " << p.first;
```

Queue (FIFO: First-In-First-Out)

Coda di elementi, simile a una coda per salire sul bus.
[coda] w $\rightarrow$ x $\rightarrow$ y $\rightarrow$ z [testa]

Dichiarazione

```
queue<int> q;
```

Metodi utili

q.push(x)	aggiunge x in coda
q.front()	legge il valore in testa
q.pop()	rimuove l'elemento in testa
q.empty()	true sse la coda è vuota

Queue

Esempio

```
/** Coda alle poste:
 * A x -> x arriva || S -> qualcuno, viene servito
 */
queue<int> q;
char operation; int client;
for(int i = 0; i < n; i++){
    cin >> operation;
    if (operation == 'A') {
        cin >> client; q.push(client);
    } else {
        if (!q.empty()){
            cout << q.front() << endl;q.pop();
        }
    }
}
```

Stack (LIFO: Last-In-First-Out)

Pila di elementi, simile a una pila di piatti.
w $\rightarrow$ x $\rightarrow$ y $\rightarrow$ z [testa]

Dichiarazione

```
stack<int> a;
```

Metodi utili

s.push(x)	aggiunge x testa
s.top()	legge il valore in testa
s.pop()	rimuove l'elemento in testa
s.empty()	true sse lo stack è vuoto

Stack

Esempio

```
/** Controllare se una stringa di parentesi è annidata co
 * Ex: {()([]{})}() OK!, {(]} NO!
 */
string p; cin >> p;
stack<int> s;
for(char c : s)
    if(c == '('  c == '['  c == '{')
        s.push(c);
    else if (!s.empty() &&
                (c == ')' && s.pop() == '('
                (c == ']' && s.pop() == '['
                (c == '}' && s.pop() == '{')
            ) s.pop();
    else return false;
return true;
```

Set

Insieme di elementi $\{w, x, y, z\} \Rightarrow$ no duplicati!

Dichiarazione

```
set<int> s; // gli elementi sono ordinati
unordered_set<int> s; // gli elementi non sono ordinati,
                      // più veloce
```

Metodi utili

s.insert(x)	aggiunge x se non c'è già
s.erase(x)	rimuove x se c'è
s.find(x)	cerca se x appartiene all'insieme

Set

Esempio

```
/** Calcolare la somma degli elementi duplicati
 */
unordered_set<int> s;
int x, sum = 0;
for(int i = 0; i < n; i++){
    cin >> x;
    // x non c'è
    if(s.find(x) == s.end()) {
        s.insert(x);
    } else {
        sum += s;
    }
}
return sum;
```

Map

Insieme di coppie chiave-valore. $\{$ key1 $\rightarrow$ value1, key2 $\rightarrow$ value2, ... $\}$
Può anche essere vista come un array dove gli indici possono non sono 0 . . . n-1, ma interi in un altro range, stringhe, ecc.

Dichiarazione

```
map<int, int> m; // Map gli elementi sono ordinati
unordered_map<string, int> m; // gli elementi non sono
                              // ordinati, più veloce
```

Metodi utili

m[x]	restituisce il valore della chiave associata a x, altrimenti 0;
m.find(x)	cerca se x appartiene alle chiavi

Map

Esempio

```
/** Data una lista di parole, contare quante volte
 * compare ognuna
 */
unordered_map<string, int> count;
string w;
for(int i = 0; i < n; i++){
    cin >> w;
    m[w]++;
}
for(pair<string, int> p : count){
    cout << p.first << " -> " << p.second << endl;
}
```