1

EDA. Exemples d'ús de la STL

1.1 Lectura línia a línia: `istringstream`

Llegeix una seqüència de línies, i de cada línia escriu la suma dels seus números.

```
#include <sstream>
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    while (getline(cin, s)) {
        istringstream ss(s);
        int suma = 0;
        int x;
        while (ss >> x) suma += x;
        cout << suma << endl;
}   }
```

1.2 Piles: `stack`

Llegeix una seqüència de números i l'escriu del revés.

```
#include <stack>
#include <iostream>
```

```
using namespace std;

int main() {
    stack<int> s;
    int x;
    while (cin >> x) s.push(x);
    while (not s.empty()) {
        cout << s.top() << endl;
        s.pop();
}   }
```

1.3 Cues: queue

Llegeix una seqüència de números i l'escriu del dret.

```
#include <queue>
#include <iostream>
using namespace std;

int main() {
    queue<int> s;
    int x;
    while (cin >> x) s.push(x);
    while (not s.empty()) {
        cout << s.front() << endl;
        s.pop();
}   }
```

1.4 Cues de prioritats: priority_queue

Llegeix una seqüència de números i l'escriu en ordre decreixent.

```
#include <queue>
#include <iostream>
using namespace std;

int main() {
    priority_queue<int> s;
    int x;
    while (cin >> x) s.push(x);
    while (not s.empty()) {
        cout << s.top() << endl;
```

```
        s.pop();
}   }
```

1.5 Cues de prioritats invertint l'ordre

(Una altra manera es posant i treient els valors canviats de signe).

Llegeix una seqüència de números i l'escriu en ordre creixent.
El terecer parametre del tipus es l'important, el segon cal omplir-lo.

```
#include <queue>
#include <iostream>
using namespace std;

int main() {
    priority_queue<int, vector<int>, greater<int>> s;
    int x;
    while (cin >> x) s.push(x);
    while (not s.empty()) {
        cout << s.top() << endl;
        s.pop();
}   }
```

1.6 Conjunts: `set`

Llegeix dues seqüències de números acabades en zero i escriu seva intersecció.

```
#include <set>
#include <iostream>
#include <string>
using namespace std;

int main() {
    set<int> s1, s2;
    int x;
    cin >> x;
    while (x != 0) {
        s1.insert(x);
        cin >> x;
    }
    cin >> x;
    while (x != 0) {
        s2.insert(x);
```

```
        cin >> x;
    }
    for (set<int>::iterator it = s1.begin(); it != s1.end(); ++it) {
        if (s2.find(*it) != s2.end()) cout << *it << endl;
}   }
```

1.7 Diccionaris: `map`

Llegeix una seqüència de paraules i, per a cada paraula, en ordre alfabètic, diu quants cops apareix.

```
#include <map>
#include <iostream>
#include <string>
using namespace std;

int main() {
    map<string, int> m;
    string x;
    while (cin >> x) ++m[x];
    for (map<string, int>::iterator it = m.begin(); it != m.end(); ++it) {
        cout << it->first << " " << it->second << endl;
}   }
```

1.8 Noves capacitats en C++11

Noves capacitats del C++11.

Amb C++11 es poden deixar al compilar que "endevini"el tipus de les variables. A més, s'ha extès el bucle `for` per tal de poder iterar fàcilment sobre les col·leccions. Ja no cal posar un espai entre els >> dels templates i es poden donar llistes d'inicialitzacions a tipus complexes.

Per compilar amb C++11, cal un GCC recent (per exemple gcc-4.7.0) i compilar amb el paràmetre `-std=c++11`.

```
#include <vector>
#include <set>
#include <iostream>
using namespace std;

int main() {
    // llegeix un vector per l'entrada
    vector<int> v;
    int x;
    while (cin >> x) v.push_back(x);
```

```
    // escriu tots els elements en v
    for (int y : v) cout << y << endl;
    cout << endl;
    // dobla tots els elements en v (fixeu-vos en la referència, per modificar els elements)
    for (int& y : v) y *= 2;
    // escriu tots els elements en v
    for (int y : v) cout << y << endl;
    cout << endl;

    // crea un conjunt de conjunts d'enters i l'escriu
    set<set<int>> S = {{2,3}, {5,1,5}, {}, {3}};
    for (auto it1 : S) {
        cout << "{";
        for (auto it2 : it1) {
            cout << it2 << ",";
        }
        cout << "}" << endl;
    }
}
```

1.9 Conjunts no ordenats: `unordered_set`

Llegeix dues seqüències de números acabades en zero i escriu seva intersecció.
Aquest codi usa C++11.

```
#include <unordered_set>
#include <iostream>
#include <string>
using namespace std;

int main() {
    unordered_set<int> s1, s2;
    int x;
    cin >> x;
    while (x != 0) {
        s1.insert(x);
        cin >> x;
    }
    cin >> x;
    while (x != 0) {
        s2.insert(x);
        cin >> x;
    }
    for (auto it : s1) {
        if (s2.find(it) != s2.end()) cout << it << endl;
}   }
```

1.10 Diccionaris no ordenats: `unordered_map`

Llegeix una seqüència de paraules i, per a cada paraula, diu quants cops apareix. Com que s'usa un map sense ordre, l'ordre de sortida no està definit.
Aquest codi usa C++11.

```
#include <unordered_map>
#include <iostream>
#include <string>
using namespace std;

int main() {
    unordered_map<string, int> m;
    string x;
    while (cin >> x) ++m[x];
    for (auto it : m) {
        cout << it.first << " " << it.second << endl;
}   }
```