

# JUTGE PRO2 FIB

## L5. Diccionaris i Conjunts

GitHub: <https://github.com/MUX-enjoyer/PRO2-FIB-2025>

### Índex de Fitxers

#### 5.1 Pairs

- X16157 Divisió i Mòdul.cc (pàgina 2)
- X18027 Llista de Parelles.cc (pàgina 3)

#### 5.2 Maps

- X10844 Suma Fruites.cc (pàgina 4)
- X34352 Freqüència de paraules amb diccionaris.cc (pàgina 5)
- X47779 Morse.cc (pàgina 6)
- X59118 Diccionari.cc (pàgina 8)
- X79905 Freqüència de paraules amb diccionaris (amb esborrat).cc (pàgina 9)
- X80993 Ok Cupid.cc (pàgina 10)

#### 5.3 Conjunts

- X51146 Activitats esportives ordenades (sets).cc (pàgina 11)
- X83904 Activitats esportives (sets).cc (pàgina 12)

## Exercici: 5.1 Pairs

X16157 Divisió i Mòdul.cc

```
1 #include <iostream>
2 #include <utility>
3 using namespace std;
4
5 pair<int, int> divmod(int a, int b) {
6     pair<int, int> parell_enters(a / b, a % b);
7     return parell_enters;
8 }
9
10 int main() {
11     int a, b;
12     cin >> a >> b;
13     pair<int, int> p = divmod(a, b);
14     cout << p.first << ' ' << p.second << endl;
15 }
```

## Exercici: 5.1 Pairs

### X18027 Llista de Parelles.cc

```
1 #include <iostream>
2 #include <list>
3 #include <map>
4 using namespace std;
5
6 list<pair<string, int>> all_pairs(map<string, int>& M) {
7 list<pair<string, int>> llista_string_int;
8
9 map<string, int>::iterator it;
10 for (it = M.begin(); it != M.end(); ++it) {
11 pair<string, int> p = *it;
12 llista_string_int.push_back(p);
13 }
14 return llista_string_int;
15 }
16
17 int main() {
18 string s;
19 int n;
20 map<string, int> M;
21 while (cin >> s >> n) {
22 M[s] = n;
23 }
24 list<pair<string, int>> L = all_pairs(M);
25 list<pair<string, int>>::iterator it = L.begin();
26 for (; it != L.end(); it++) {
27 cout << it->first << ' ' << it->second << endl;
28 }
29 }
```

## Exercici: 5.2 Maps

X10844 Suma Fruites.cc

```
1 #include <iostream>
2 #include <map>
3 using namespace std;
4
5 int main() {
6     map<string, int> fruites;
7     string fruita;
8     int quantitat;
9
10    while (cin >> fruita >> quantitat) {
11        fruites[fruita] += quantitat;
12    }
13
14    map<string, int>::iterator it;
15    for (it = fruites.begin(); it != fruites.end(); ++it) {
16        cout << it->first << " " << it->second << endl;
17    }
18 }
```

## Exercici: 5.2 Maps

X34352 Freqüència de paraules amb diccionaris.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <map>
6
7 int main() {
8     map<string, int> partits;
9     string codi, paraula;
10    while(cin >> codi >> paraula) {
11        if (codi == "a") ++partits[paraula];
12        else if (codi == "f") cout << partits[paraula] << endl;
13    }
14 }
```

## Exercici: 5.2 Maps

X47779 Morse.cc

```
1 #include <iostream>
2 #include <map>
3 using namespace std;
4
5 string to_morse(string s, const map<char, string>& M) {
6     string morse;
7     for (int i = 0; i < s.size(); ++i) {
8         char lletra = toupper(s[i]);
9         if (lletra != ' ') {
10             if (i != 0) morse.push_back(' ');
11             morse.append(M.at(lletra));
12         }
13     }
14     return morse;
15 }
16
17 int main() {
18     map<char, string> M;
19     M['A'] = ".-";
20     M['B'] = "-...";
21     M['C'] = "-.-.";
22     M['D'] = "-..";
23     M['E'] = ".";
24     M['F'] = "..-.";
25     M['G'] = "--.";
26     M['H'] = "....";
27     M['I'] = "..";
28     M['J'] = ".---";
29     M['K'] = "-.-";
30     M['L'] = ".-..";
31     M['M'] = "--";
32     M['N'] = "-.";
33     M['O'] = "---";
34     M['P'] = ".-.-";
35     M['Q'] = "--.-";
36     M['R'] = ".-.";
37     M['S'] = "...";
38     M['T'] = "-";
39     M['U'] = "...-";
40     M['V'] = "...-";
41     M['W'] = ".-.-";
42     M['X'] = "-.-.-";
43     M['Y'] = "-.-.-";
44     M['Z'] = "--..";
45     M['0'] = "-----";
46     M['1'] = ".-----";
47     M['2'] = "..-----";
48     M['3'] = "...-----";
49     M['4'] = "....-";
50     M['5'] = ".....";
51     M['6'] = "-.....";
52     M['7'] = "--...";
53     M['8'] = "---..";
54     M['9'] = "----.";
55 }
```

```
56 string s;  
57 getline(cin, s);  
58 cout << to_morse(s, M) << endl;  
59 }
```

## Exercici: 5.2 Maps

X59118 Diccioniari.cc

```
1 #include <iostream>
2 #include <map>
3 #include <sstream>
4
5 using namespace std;
6
7 /** * Pre: L'entrada conté una seqüència de parells (paraula_A, paraula_B)
8 seguit d'una línia buida, * després una seqüència de paraules en l'idioma A a
9 traduir. * Post: Es mostra la traducció de cada paraula en l'idioma B o "<not
10 found>" si no es troba en el diccionari. */
11
12 int main() {
13     map<string, string> diccionari;
14     string linia, paraula_A, paraula_B;
15
16     // Llegir la primera part (diccionari) while (getline(cin, linia) &&
17     !linia.empty()) {
18         istringstream ss(linia);
19         ss >> paraula_A >> paraula_B;
20         diccionari[paraula_A] = paraula_B;
21     }
22
23     // Llegir la segona part (paraules a traduir) while (cin >> paraula_A) {
24     if (diccionari.count(paraula_A)) cout << diccionari[paraula_A] << endl;
25     else cout << "<not found>" << endl;
26 }
27 }
```



## Exercici: 5.2 Maps

X79905 Freqüència de paraules amb diccionaris (amb esborrat).cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <map>
6
7 int main() {
8     map<string, int> partits;
9     string codi, paraula;
10    while(cin >> codi >> paraula) {
11        if (codi == "a") ++partits[paraula];
12        else if (codi == "e" && partits[paraula] > 0) --partits[paraula];
13        else if (codi == "f") cout << partits[paraula] << endl;
14    }
15 }
```

## Exercici: 5.2 Maps

X80993 Ok Cupid.cc

```

1 #include <iostream>
2 #include <map>
3 #include <vector>
4
5 using namespace std;
6
7 /** * Pre: L'entrada consisteix en una seqüència de parelles (nom, t), on `nom`
és un string sense espais i `t` és un enter dins del rang  $[-10^6, 10^6]$ . * Post:
Els usuaris i els seus valors s'han emmagatzemat a `usuaris` i `per_valor` segons
el seu valor t. */
8 void llegir_usuaris(map<string, int>& usuaris, map<int, vector<string>>&
per_valor) {
9     string nom;
10    int t;
11    while (cin >> nom >> t) {
12        usuaris[nom] = t;
13        per_valor[t].push_back(nom);
14    }
15 }
16
17 /** * Pre: `usuaris` conté una llista d'usuaris amb els seus valors t, i
`per_valor` agrupa usuaris segons aquests valors. * Post: Es mostra una línia per
usuari amb els seus compatibles, en ordre alfabètic. */
18 void trobar_compatibles(const map<string, int>& usuaris, const map<int,
vector<string>>& per_valor) {
19     map<string, int>::const_iterator it;
20     for (it = usuaris.begin(); it != usuaris.end(); ++it) {
21         const string& usuari = it->first;
22         int valor = it->second;
23         int oposat = -valor;
24
25         cout << usuari;
26
27         if (per_valor.count(oposat)) {
28             vector<string>::const_iterator it2;
29             for (it2 = per_valor.at(oposat).begin(); it2 != per_valor.at(oposat).end();
++it2) {
30                 if (*it2 != usuari) cout << " " << *it2;
31             }
32         }
33         cout << endl;
34     }
35 }
36
37 /** * Pre: L'entrada segueix el format indicat. * Post: Es llegeixen els
usuaris i es mostra la seva compatibilitat. */
38 int main() {
39     map<string, int> usuaris;
40     map<int, vector<string>> per_valor;
41
42     llegir_usuaris(usuaris, per_valor);
43     trobar_compatibles(usuaris, per_valor);
44
45 }
```

## Exercici: 5.3 Conjunts

X51146 Activitats esportives ordenades (sets).cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <set>
6 #include <map>
7
8 int main() {
9     set<string> cap_act;
10    map<string, int> assistencia_act;
11
12    string cognom;
13    cin >> cognom;
14    while (cognom != ".") {
15        cap_act.insert(cognom);
16        cin >> cognom;
17    }
18
19    int n;
20    cin >> n;
21    for (int i = 0; i < n; ++i) {
22        cin >> cognom;
23        while (cognom != ".") {
24            cap_act.erase(cognom);
25            ++assistencia_act[cognom];
26            cin >> cognom;
27        }
28    }
29
30    cout << "Totes les activitats:";
31    map<string, int>::iterator it_tot;
32    for (it_tot = assistencia_act.begin(); it_tot != assistencia_act.end();
33    ++it_tot) {
34        if (it_tot->second == n) cout << ' ' << it_tot->first;
35    }
36    cout << endl << "Cap activitat:";
37    set<string>::iterator it_cap;
38    for (it_cap = cap_act.begin(); it_cap != cap_act.end(); ++it_cap) {
39        cout << ' ' << *it_cap;
40    }
41    cout << endl;
42 }
```

## Exercici: 5.3 Conjunts

### X83904 Activitats esportives (sets).cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <set>
6
7 int main() {
8     set<string> activitat, totes_activitats, cap_activitat;
9     set<string>::iterator it_act, it_tot, it_cap;
10
11     string cognom;
12     cin >> cognom;
13     while (cognom != ".") {
14         cap_activitat.insert(cognom);
15         cin >> cognom;
16     }
17
18     int n;
19     cin >> n;
20     totes_activitats = cap_activitat;
21     for (int i = 0; i < n; ++i) {
22         cin >> cognom;
23         while (cognom != ".") {
24             activitat.insert(cognom);
25             cin >> cognom;
26         }
27         for (it_act = activitat.begin(); it_act != activitat.end(); ++it_act) {
28             it_cap = cap_activitat.find(*it_act);
29             if (it_cap != cap_activitat.end()) cap_activitat.erase(it_cap);
30         }
31         for (it_tot = totes_activitats.begin(); it_tot != totes_activitats.end(); ++it_tot) {
32             it_act = activitat.find(*it_tot);
33             if (it_act == activitat.end()) it_tot = totes_activitats.erase(it_tot);
34             else ++it_tot;
35         }
36         activitat.clear();
37     }
38
39     cout << "Totes les activitats:";
40     for (it_tot = totes_activitats.begin(); it_tot != totes_activitats.end(); ++it_tot) {
41         cout << ' ' << *it_tot;
42     }
43     cout << endl << "Cap activitat:";
44     for (it_cap = cap_activitat.begin(); it_cap != cap_activitat.end(); ++it_cap) {
45         cout << ' ' << *it_cap;
46     }
47     cout << endl;
48 }
```