

JUTGE PRO2 FIB

L4. Iteracions

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

Índex de Fitxers

4.1 Iteradors

- X31697 Don Camilo.cc (pàgina 2)
- X31852 Ajuntar paraules.cc (pàgina 3)
- X80667 Guanyadors.cc (pàgina 4)
- X83553 Paraula més llarga.cc (pàgina 5)

4.2 Llistes

- X27494 Estadístiques d'una seqüència d'enters amb esborrat.cc (pàgina 15)
- X60047 Punt mig d'una llista.cc (pàgina 17)
- X87360 Intersecció de llistes ordenades.cc (pàgina 18)
- **X19134 Cerca en una llista de parells d'enters**
 - LlistalOParInt.cc (pàgina 6)
 - LlistalOParInt.hh (pàgina 7)
 - ParInt.cc (pàgina 8)
 - ParInt.hh (pàgina 9)
 - program.cc (pàgina 10)
- **X21215 Implementació de matrius esparses**
 - PRO2Excepcio.hh (pàgina 11)
 - matriu.hh (pàgina 12)
 - program.cc (pàgina 14)

Exercici: 4.1 Iteradors

X31697 Don Camilo.cc

```
1 #include <iostream>
2 #include <vector>
3 #include <iterator>
4 using namespace std;
5
6 void inserta_don(vector<string>& frase) {
7     vector<string>::iterator it;
8     for (it = frase.begin(); it != frase.end(); ++it) {
9         if (*it == "Camilo") {
10             it = frase.insert(it, "Don");
11             ++it;
12         }
13     }
14 }
15
16 int main() {
17     string p;
18     vector<string> frase;
19     while (cin >> p) frase.push_back(p);
20     inserta_don(frase);
21     copy(frase.begin(), frase.end(),
22         ostream_iterator<string>(cout, " "));
23     cout << endl;
24 }
```

Exercici: 4.1 Iteradors

X31852 Ajuntar paraules.cc

```
1 #include <iostream>
2 #include <sstream>
3 #include <vector>
4 using namespace std;
5
6 string join(vector<string>::iterator ini, vector<string>::iterator fin, string
sep) {
7     string s;
8     if (ini == fin) return s;
9     vector<string>::iterator it = ini;
10    s.append(*it);
11    for (++it; it != fin; ++it) {
12        s.append(sep);
13        s.append(*it);
14    }
15    return s;
16 }
17
18 int main() {
19     string line;
20     getline(cin, line);
21     const string separator = line.substr(1, line.size() - 2);
22
23     getline(cin, line);
24     istringstream S(line);
25
26     vector<string> words;
27     string word;
28     while (S >> word) {
29         words.push_back(word);
30     }
31
32     cout << '"' << join(words.begin(), words.end(), separator) << '"' << endl;
33
34     return 0;
35 }
```

Exercici: 4.1 Iteradors

X80667 Guanyadors.cc

```
1 #include <iostream>
2 #include <iterator>
3 #include <string>
4 #include <vector>
5 using namespace std;
6
7 struct Player {
8     string name;
9     int score;
10 };
11
12 vector<string> winners(vector<Player>::iterator ini, vector<Player>::iterator
fin) {
13     vector<string> winners;
14     if (ini == fin) return winners;
15
16     int max_punt = ini->score;
17     for (vector<Player>::iterator it = ini; it != fin; ++it) {
18         if (it->score == max_punt) {
19             winners.push_back(it->name);
20         } else if (it->score > max_punt) {
21             max_punt = it->score;
22             winners.clear();
23             winners.push_back(it->name);
24         }
25     }
26     return winners;
27 }
28
29 int main() {
30     Player p;
31     vector<Player> v;
32     while (cin >> p.name >> p.score) {
33         v.push_back(p);
34     }
35     vector<string> w = winners(v.begin(), v.end());
36     for (int i = 0; i < w.size(); i++) {
37         cout << (i == 0 ? "" : " ") << w[i];
38     }
39     cout << endl;
40 }
```

Exercici: 4.1 Iteradors

X83553 Paraula més llarga.cc

```
1 #include <iostream>
2 #include <list>
3 #include <string>
4 using namespace std;
5
6 list<string>::const_iterator longest(const list<string>& v) {
7     int mida_max = 0;
8     list<string>::const_iterator longest_word = v.end();
9     list<string>::const_iterator it;
10    for (it = v.begin(); it != v.end(); ++it) {
11        string paraula = *it;
12        if (paraula.size() > mida_max) {
13            mida_max = paraula.size();
14            longest_word = it;
15        }
16    }
17    return longest_word;
18 }
19
20 int main() {
21     string word;
22     list<string> text;
23     while (cin >> word) {
24         text.push_back(word);
25     }
26     auto i = longest(text);
27     if (i != text.end()) {
28         cout << *i << endl;
29     } else {
30         cout << "<empty sequence>" << endl;
31     }
32     return 0;
33 }
```

Exercici: X19134 Cerca en una llista de parells d'enters

LlistaOParInt.cc

```
1 using namespace std;
2 #include <iostream>
3
4 #include <list>
5
6 #include "ParInt.hh"
7
8 void LlegirLlistaParInt(list<ParInt>& l) {
9     ParInt parell;
10    while (parell.llegir()) l.push_front(parell);
11 }
```

Exercici: X19134 Cerca en una llista de parells d'enters

LlistaOParInt.hh

```
1 void LlegirLlistaParInt(list<ParInt>& l);  
2 // Pre: l és buida; el canal estandar d'entrada conté un nombre // parell  
d'enters, acabat pel parell 0 0 // Post: s'han afegit al final de l els parells  
llegits fins al 0 0 (no inclòs)  
3 void EscriureLlistaParInt(const list<ParInt>& l); //(opcional) // Pre: cert //  
Post: s'han escrit al canal estandar de sortida els elements de l
```

Exercici: X19134 Cerca en una llista de parells d'enters

ParInt.cc

```
1 #include "ParInt.hh"
2
3 ParInt::ParInt(){
4     p=0;s=0;
5 }
6
7 ParInt::ParInt(int a,int b){
8     p=a;s=b;
9 }
10
11 int ParInt::primer() const{
12     return p;
13 }
14
15 int ParInt::segon() const{
16     return s;
17 }
18
19 bool ParInt::llegir(){
20     cin >> p >> s;
21     return (p!=0 or s!=0);
22 }
23
24 void ParInt::escriure() const{
25     cout << p << ' ' << s << endl;
26 }
```


Exercici: X19134 Cerca en una llista de parells d'enters

ParInt.hh

```
1 #ifndef CLASS_ParInt_HH
2 #define CLASS_ParInt_HH
3
4 #include <iostream>
5 using namespace std;
6
7 class ParInt {
8
9 private:
10
11 int p;
12 int s;
13
14 public:
15
16 //Constructores
17 ParInt();
18 /* Pre: cert */
19 /* Post: el resultat es el parint (0,0) */
20 ParInt(int a,int b);
21 /* Pre: cert */
22 /* Post: el resultat es el parint (a,b) */
23
24 //Consultores
25 int primer() const;
26 /* Pre: cert*/
27 /* Post: retorna el valor de p */
28 int segon() const;
29 /* Pre: cert*/
30 /* Post: retorna el valor de s */
31
32 //Entrada/sortida
33 bool llegir();
34 /* Pre: cert*/
35 /* Post: llegeix dos enters i els assigna al parametre implicit, a mes, */
36 /* si llegeix el parell 0 0 retorna fals, altrament retorna cert */
37
38 void escriure() const;
39 /* Pre: cert */
40 /* Post: escriu el parametre implicit per la sortida estandard */
41
42 };
43 #endif
```

Exercici: X19134 Cerca en una llista de parells d'enters

program.cc

```
1 using namespace std;
2 #include <iostream>
3
4 #include <list>
5
6 #include "ParInt.hh"
7 #include "LlistaIOParInt.hh"
8
9 ParInt BuscarLlista(list<ParInt> llista, int n) {
10     int repe = 0;
11     int suma = 0;
12     while (!llista.empty()) {
13         if (llista.front().primer() == n) {
14             ++repe;
15             suma += llista.front().segon();
16         }
17         llista.pop_front();
18     }
19     return ParInt(repe, suma);
20 }
21
22
23 int main() {
24     list<ParInt> llista;
25     LlegirLlistaParInt(llista);
26     int n;
27     cin >> n;
28     ParInt resultats = BuscarLlista(llista, n);
29     cout << n << ' ' << resultats.primer() << ' ' << resultats.segon() << endl;
30 }
```

Exercici: X21215 Implementació de matrius esparses

PRO2Excepcio.hh

```
1 #ifndef PRO2_EXCEPCIO // per evitar fer mes d'un include d'aquest fitxer
2 #define PRO2_EXCEPCIO
3 #include <exception>
4
5 using namespace std;
6
7 /* Classe PRO2Excepcio */
8 class PRO2Excepcio: public exception {
9 public:
10 PRO2Excepcio(const char* mot) : exception(), mensaje(mot) {}
11 const char* what() const throw() {return mensaje;};
12 private:
13 const char* mensaje;
14 };
15 /* Fi classe PRO2Excepcio*/
16
17 #endif
```

Exercici: X21215 Implementació de matrius esparses

matriu.hh

```

1 #include <iostream>
2 #include <list>
3 #include <vector>
4
5 #include "PRO2Excepcio.hh"
6
7 using namespace std;
8
9 template <typename T>
10 class matriu {
11 private:
12 struct parint {
13 int col;
14 T val;
15 };
16
17 vector<list<parint> > files; // elements de la matriu int nfil; // dimensions
de la matriu int ncol;
18
19 // Representaci  de matrius amb implementaci  dispersa // (nom s desem els
valors diferents de zero). // Les files es desen en un vector de parells tals que
// per al parell (j,x) de la fila i: // // x != 0 <=> m[i][j] = x <=> (j,x) \in
files[i] // // Els parells de la llista estan ordenats creixentment pel primer
element del parell.
20 public:
21 // Constructores
22 matriu(int m, int n);
23 /* Pre: n>0 */
24 /* Post: matriu de mXn amb tots els elements a 0 */
25
26 // Modificadores
27 void modif_pos(int i, int j, T x);
28 /* Pre: 0<=i< num files del par metre impl cit; 0<=j< num columnes del
par metre impl cit */
29 /* Post: el par metre impl cit queda com l'original per  amb x a la la
posici  [i][j] */
30
31 void suma(const matriu& m1, const matriu& m2);
32 /* Pre: m1 i m2 tenen la mateixa dimensi  */
33 /* Post: el par metre impl cit  s la suma de m1 i m2 */
34
35 void producte(const matriu& m1, const matriu& m2);
36 /* Pre: el n mero de columnes de m1  s igual al n mero de files de m2 */
37 /* Post: el par metre impl cit  s el producte de m1 i m2 */
38
39 // Consultores
40 int num_files() const;
41 /* Pre: cert */
42 /* Post: El resultat es el n mero de files del par metre impl cit */
43
44 int num_columnes() const;
45 /* Pre: cert */
46 /* Post: El resultat es el n mero de columnes del par metre impl cit */
47
48 T pos(int i, int j) const;

```

```
49 /* Pre: 0<=i< num files del par■metre impl■cit; 0<=j< num columnes del
par■metre impl■cit */
50 /* Post: El resultat ■s l'element [i][j] del par■metre impl■cit */
51
52 // Lectura i escriptura.
53 void llegeix(int m, int n);
54 /* Pre: el canal est■ndard d'entrada t■ parells d'enters que representen una
matriu mXn representada de forma espar■a; al principi de cada fila apareix el seu
n■mero de valors diferents de 0 */
55
56 /* Post: El parametre impl■cit t■ la matriz formada per parells del canal
est■ndard d'entrada original; si m o n no coincideixen amb les dimensions del
p.i. el qual queda redimensionat */
57
58 void escriu() const;
59 /* Pre: cert */
60 /* Post: La sortida est■ndard t■ els valors de la matriu de forma can■nica
(valors fila per fila). */
61 };
```

Exercici: X21215 Implementació de matrius esparses

program.cc

```
1 #include "matriu.tcc"
2
3 int main() {
4
5     int operacio;
6     cin >> operacio;
7
8     matriu <int> m1(0,0);
9     int nf1, nc1;
10    cin >> nf1 >> nc1;
11    m1.llegeix (nf1,nc1);
12
13    int nf2, nc2;
14    cin >> nf2 >> nc2;
15
16    matriu <int> m2(0,0);
17    m2.llegeix(nf2,nc2);
18
19    m1.escriu();
20    m2.escriu();
21
22    if (operacio == 1) {
23        matriu <int> s(0,0);
24
25        s.suma(m1,m2);
26        s.escriu();
27    }
28
29    if (operacio == 2) {
30        matriu <int> p(0,0);
31
32        p.producte(m1,m2);
33        p.escriu();
34    }
35
36 }
```

Exercici: 4.2 Llistes

X27494 Estadístiques d'una seqüència d'enters amb esborrat.cc

```
1 using namespace std;
2 #include <iostream>
3 #include <list>
4
5 void min_max(const list<int>& l, int& min, int& max) {
6     list<int>::const_iterator it = l.begin();
7     min = max = *it;
8     for (++it; it != l.end(); ++it) {
9         if (min > *it)
10             min = *it;
11         else if (max < *it)
12             max = *it;
13     }
14 }
15
16 void borra(list<int>& llista, float& suma, int n, int& size) {
17     for (list<int>::iterator it = llista.begin(); it != llista.end(); ++it) {
18         if (*it == n) {
19             llista.erase(it);
20             --size;
21             suma -= n;
22             return;
23         }
24     }
25 }
26
27 int main() {
28     list<int> llista;
29     int codi, n, min, max;
30     float suma = 0;
31     int size = 0;
32     cin >> codi >> n;
33     min = max = n;
34     while (n != 0 || codi != 0) {
35         if (codi == -1) {
36             if (size == 0) {
37                 min = n;
38                 max = n;
39             } else if (n < min)
40                 min = n;
41             else if (n > max)
42                 max = n;
43             llista.push_back(n);
44             ++size;
45             suma += n;
46         } else if (!llista.empty() && codi == -2) {
47             borra(llista, suma, n, size);
48             if (!llista.empty() && (min == n || max == n)) min_max(llista, min, max);
49         }
50     }
51     if (llista.empty()) {
52         cout << 0 << endl;
53         suma = 0;
54     } else
55 }
```

```
56 cout << min << ' ' << max << ' ' << suma / size << endl;  
57 cin >> codi >> n;  
58 }  
59 }
```


Exercici: 4.2 Llistes

X60047 Punt mig d'una llista.cc

```
1 using namespace std;
2 #include <iostream>
3 #include <list>
4 #include <string>
5
6 int main() {
7     list<int> llista;
8     list<int>::iterator mid = llista.end();
9     int x, size = 0;
10    string instruccio;
11
12    while (cin >> instruccio) {
13        if (instruccio == "get_mid_value") {
14            if (size % 2 == 0) cout << "error" << endl;
15            else cout << *mid << endl;
16        } else if (instruccio == "push_front") {
17            cin >> x;
18            llista.push_front(x);
19            ++size;
20            if (size == 1) mid = llista.begin();
21            else if (size % 2 == 1) --mid;
22        } else if (instruccio == "push_back") {
23            cin >> x;
24            llista.push_back(x);
25            ++size;
26            if (size == 1) mid = llista.begin();
27            else if (size % 2 == 0) ++mid;
28        } else if (instruccio == "pop_front") {
29            if (llista.empty()) cout << "error" << endl;
30            else {
31                if (size % 2 == 1) ++mid;
32                llista.pop_front();
33                --size;
34                if (size == 0) mid = llista.end();
35            }
36        } else if (instruccio == "pop_back") {
37            if (llista.empty()) cout << "error" << endl;
38            else {
39                if (size % 2 == 0) --mid;
40                llista.pop_back();
41                --size;
42                if (size == 0) mid = llista.end();
43            }
44        }
45    }
46 }
```

Exercici: 4.2 Llistes

X87360 Intersecció de llistes ordenades.cc

```
1 using namespace std;
2 #include <iostream>
3
4 #include <list>
5
6 void inter(list<int>& uno, const list<int>& dos ) {
7     /* Pre: uno = U */
8     /* Post: uno pasa a ser la interseccion de U y dos */
9     list<int>::iterator it1 = uno.begin();
10    list<int>::const_iterator it2 = dos.begin();
11    while (it1 != uno.end() && it2 != dos.end()) {
12        if (*it1 < *it2) it1 = uno.erase(it1);
13        else if (*it1 > *it2) ++it2;
14        else {
15            ++it1;
16            ++it2;
17        }
18    }
19    // Elimina els elements sobrants a la fi de uno (si n'hi ha) uno.erase(it1,
20    uno.end());
21 }
```