

JUTGE PRO2 FIB

L3. Piles i Cues

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

Índex de Fitxers

3.1 Piles

---- X36902 Avaluacio d'una expressio amb parentesis.cc (pàgina 2)

---- X68213 Biblioteca.cc (pàgina 3)

---- X80203 Indexar seqüències ben parentitzades.cc (pàgina 4)

---- X96935 Palíndroms amb piles.cc (pàgina 5)

3.2 Cues

---- P90861 Cues d'un supermercat (1).cc (pàgina 6)

---- X38371 Estadístiques d'una seqüència d'enters amb esborrat del més antic.cc (pàgina 12)

---- **X13425 Distribucio justa de cues**

----- CualOParInt.cc (pàgina 7)

----- CualOParInt.hh (pàgina 8)

----- ParInt.cc (pàgina 9)

----- ParInt.hh (pàgina 10)

----- program.cc (pàgina 11)

Exercici: 3.1 Piles

X36902 Avaluacio d'una expressio amb parentesis.cc

```
1 #include <iostream>
2 using namespace std;
3 #include <stack>
4
5 int main() {
6     char x;
7     int pos = 0, i = 1;
8     stack<char> pila;
9     cin >> x;
10    while(x != '.') {
11        if ((x == '(' || x == '[')) pila.push(x);
12        else if (!pila.empty()) {
13            if (pila.top() == '(' && x == ')') pila.pop();
14            else if (pila.top() == '[' && x == ']') pila.pop();
15            else if (pos == 0) pos = i;
16        }
17        else if (pos == 0) pos = i;
18        ++i;
19        cin >> x;
20    }
21    if (pos != 0) cout << "Incorrecte " << pos << endl;
22    else if (!pila.empty()) cout << "Incorrecte " << i-1 << endl;
23    else cout << "Correcte" << endl;
24 }
```

Exercici: 3.1 Piles

X68213 Biblioteca.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <vector>
6 #include <stack>
7
8 int main() {
9     int n, opcio;
10    char x;
11    cin >> n >> x >> opcio;
12    vector<stack<string>> piles(n);
13
14    while (opcio != 4) {
15        int num_pila;
16        if (opcio == 1) {
17            string llibre;
18            cin >> llibre >> num_pila;
19            piles[num_pila-1].push(llibre);
20        }
21        if (opcio == 2) {
22            int n_llibres;
23            cin >> n_llibres >> num_pila;
24            for (int i = 0; i < n_llibres; ++i) {
25                piles[num_pila-1].pop();
26            }
27        }
28        if (opcio == 3) {
29            cin >> num_pila;
30            stack<string> copia_pila = piles[num_pila-1];
31            cout << "Pila de la categoria " << num_pila << endl;
32            for (int i = 0; !copia_pila.empty(); ++i) {
33                cout << copia_pila.top() << endl;
34                copia_pila.pop();
35            }
36            cout << endl;
37        }
38
39        cin >> x >> opcio;
40    }
41 }
```

Exercici: 3.1 Piles

X80203 Indexar seqüències ben parentitzades.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <string>
5 #include <stack>
6
7 int main() {
8     string x;
9     stack<int> pila;
10    while(cin >> x) {
11        int mida = x.size();
12        int cont = 1;
13        for (int i = 0; i < mida; ++i) {
14            if (x[i] == '(') {
15                pila.push(cont);
16                cout << x[i] << pila.top();
17                ++cont;
18            }
19            else if (x[i] == ')') {
20                cout << x[i] << pila.top();
21                pila.pop();
22            }
23        }
24        cout << endl;
25    }
26 }
```

Exercici: 3.1 Piles

X96935 Palíndroms amb piles.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <stack>
5
6 bool palindrom(stack<int> s, int n) {
7     int x;
8     if (n%2 != 0) cin >> x;
9     for (int i = 0; i < n/2; ++i) {
10         cin >> x;
11         if (s.top() != x) return false;
12         s.pop();
13     }
14     return true;
15 }
16
17 int main() {
18     int n;
19     stack<int> nums;
20     cin >> n;
21     for (int i = 0; i < n/2; ++i) {
22         int a;
23         cin >> a;
24         nums.push(a);
25     }
26     if(palindrom(nums, n)) cout << "SI" << endl;
27     else cout << "NO" << endl;
28 }
```

Exercici: 3.2 Cues

P90861 Cues d'un supermercat (1).cc

```

1 #include <iostream>
2 using namespace std;
3
4 #include <queue>
5 #include <sstream>
6 #include <string>
7 #include <vector>
8
9 int main() {
10     int n;
11     cin >> n;
12     string x;
13     cin.ignore(); // Ignora el salt de linea buit després de llegir 'n'
14     vector<queue<string>> cues(n);
15
16     for (int i = 0; i < n; ++i) {
17         string line;
18         getline(cin, line); // Agafem tota la linea sencera i la posem a line
19         stringstream ss(line); // Dividim line en paraules dins ss while (ss >> x) {
20         cues[i].push(x);
21     }
22
23     cout << "SORTIDES" << endl
24     << "-----" << endl;
25     while (cin >> x) {
26         if (x == "SURT") {
27             int cua;
28             cin >> cua;
29             if (cua <= n && cua >= 1 && !cues[cua - 1].empty()) {
30                 cout << cues[cua - 1].front() << endl;
31                 cues[cua - 1].pop();
32             }
33             else if (x == "ENTRA") {
34                 string nom;
35                 int cua;
36                 cin >> nom >> cua;
37                 if (cua <= n && cua >= 1) cues[cua - 1].push(nom);
38             }
39         }
40
41         cout << endl
42         << "CONTINGUTS FINALS" << endl
43         << "-----" << endl;
44         for (int i = 1; i <= n; ++i) {
45             cout << "cua " << i << ':';
46             while (!cues[i - 1].empty()) {
47                 cout << ' ' << cues[i - 1].front();
48                 cues[i - 1].pop();
49             }
50             cout << endl;
51         }
52     }

```

Exercici: X13425 Distribucio justa de cues

CualOParInt.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <queue>
5
6 #include "ParInt.hh"
7
8 void llegirCuaParInt(queue<ParInt>& c) {
9     ParInt parell_enters;
10    while(parell_enters.llegir()) {
11        c.push(parell_enters);
12    }
13 }
14
15 void escriureCuaParInt(queue<ParInt> c) {
16    while(!c.empty()) {
17        ParInt parell_enters = c.front();
18        parell_enters.escriure();
19        c.pop();
20    }
21 }
```

Exercici: X13425 Distribucio justa de cues

CualOParInt.hh

```
1 #ifndef CLASS_ParInt_HH
2 #define CLASS_ParInt_HH
3
4 #include <iostream>
5 using namespace std;
6 #include "ParInt.hh"
7
8 void llegirCuaParInt(queue<ParInt>& c);
9 // Pre: c és buida; el canal estandar d'entrada conté un nombre // parell
d'enters, acabat pel parell 0 0 // Post: s'han encuat a c els elements llegits
fins al 0 0 (no inclòs)
10 void escriureCuaParInt(queue<ParInt> c);
11 // Pre: cert // Post: s'han escrit al canal estandar de sortida els elements
de c
12 #endif
```


Exercici: X13425 Distribucio justa de cues

ParInt.cc

```
1 #include "ParInt.hh"
2
3
4 ParInt::ParInt(){
5     p=0;s=0;
6 }
7
8 ParInt::ParInt(int a,int b){
9     p=a;s=b;
10 }
11
12 int ParInt::primer() const{
13     return p;
14 }
15
16 int ParInt::segon() const{
17     return s;
18 }
19
20 bool ParInt::llegir(){
21     cin >> p >> s;
22     return (p!=0 or s!=0);
23 }
24
25 void ParInt::escriure(){
26     cout << p << ' ' << s << endl;
27 }
```

Exercici: X13425 Distribucio justa de cues

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 void escriure();
38 /* Pre: cert */
39 /* Post: escriu el parametre implicit per la sortida estandard */
40
41 };
42 #endif
```

Exercici: X13425 Distribucio justa de cues

program.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <queue>
5 #include <string>
6
7 #include "CuaIOParInt.hh"
8 #include "ParInt.hh"
9
10 int main() {
11     queue<ParInt> cua_0, cua_1, cua_2;
12     llegirCuaParInt(cua_0);
13     int espera_1 = 0;
14     int espera_2 = 0;
15
16     while (!cua_0.empty()) {
17         if (espera_1 <= espera_2) {
18             cua_1.push(cua_0.front());
19             espera_1 += cua_0.front().segon();
20             cua_0.pop();
21         } else {
22             cua_2.push(cua_0.front());
23             espera_2 += cua_0.front().segon();
24             cua_0.pop();
25         }
26     }
27     escriureCuaParInt(cua_1);
28     cout << endl;
29     escriureCuaParInt(cua_2);
30 }
```

Exercici: 3.2 Cues

X38371 Estadístiques d'una seqüència d'enters amb esborrat del més antic.cc

```
1 #include <iostream>
2 using namespace std;
3
4 #include <queue>
5
6 void min_max(queue<int> c, int& min, int& max) {
7     min = max = c.front();
8     c.pop();
9     while (!c.empty()) {
10         if (c.front() < min) min = c.front();
11         else if (c.front() > max) max = c.front();
12         c.pop();
13     }
14 }
15
16 int main() {
17     queue<int> cua;
18     int n, min, max;
19     cin >> n;
20     double suma = 0;
21     min = max = n;
22     while (n <= 1000 && n >= -1001) {
23         if (n != -1001) {
24             if (n < min) min = n;
25             else if (n > max) max = n;
26             suma += n;
27             cua.push(n);
28         }
29         else if (!cua.empty()) {
30             int num_abans = cua.front();
31             suma -= num_abans;
32             cua.pop();
33             if (!cua.empty() && (min == num_abans || max == num_abans)) min_max(cua, min,
max);
34         }
35
36         if (cua.empty()) {
37             int min = 1001;
38             int max = -1001;
39             suma = 0;
40             cout << 0;
41         }
42         else cout << "min " << min << "; max " << max << "; media " <<
suma/cua.size();
43         cout << endl;
44         cin >> n;
45     }
46 }
```