JUTGE PRO2 FIB

L4. Iteracions

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

Índex de Fitxers

Exercici: 4.1 Iteradors

X31697 Don Camilo.cc

```cpp
#include <iostream>
#include <vector>
#include <iterator>
using namespace std;

void inserta_don(vector<string>& frase) {
vector<string>::iterator it;
for (it = frase.begin(); it != frase.end(); ++it) {
if (*it == "Camilo") {
it = frase.insert(it, "Don");
++it;
}
}
}

int main() {
string p;
vector<string> frase;
while (cin >> p) frase.push_back(p);
inserta_don(frase);
copy(frase.begin(), frase.end(),
ostream_iterator<string>(cout, " "));
cout << endl;
}
```

Exercici: 4.1 Iteradors

X31852 Ajuntar paraules.cc

```cpp
#include <iostream>
#include <sstream>
#include <vector>
using namespace std;

string join(vector<string>::iterator ini, vector<string>::iterator fin, string sep) {
string s;
if (ini == fin) return s;
vector<string>::iterator it = ini;
s.append(*it);
for (++it; it != fin; ++it) {
s.append(sep);
s.append(*it);
}
return s;
}

int main() {
string line;
getline(cin, line);
const string separator = line.substr(1, line.size() - 2);

getline(cin, line);
istringstream S(line);

vector<string> words;
string word;
while (S >> word) {
words.push_back(word);
}

cout << '"' << join(words.begin(), words.end(), separator) << '"' << endl;

return 0;
}
```

Mux_Enjoyer

Exercici: 4.1 Iteradors

X80667 Guanyadors.cc

```cpp
#include <iostream>
#include <iterator>
#include <string>
#include <vector>
using namespace std;

struct Player {
string name;
int score;
};

vector<string> winners(vector<Player>::iterator ini, vector<Player>::iterator
fin) {
vector<string> winners;
if (ini == fin) return winners;

int max_punt = ini->score;
for (vector<Player>::iterator it = ini; it != fin; ++it) {
if (it->score == max_punt) {
winners.push_back(it->name);
} else if (it->score > max_punt) {
max_punt = it->score;
winners.clear();
winners.push_back(it->name);
}
}
return winners;
}

int main() {
Player p;
vector<Player> v;
while (cin >> p.name >> p.score) {
v.push_back(p);
}
vector<string> w = winners(v.begin(), v.end());
for (int i = 0; i < w.size(); i++) {
cout << (i == 0 ? "" : " ") << w[i];
}
cout << endl;
}
```

Exercici: 4.1 Iteradors

X83553 Paraula més llarga.cc

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

list<string>::const_iterator longest(const list<string>& v) {
int mida_max = 0;
list<string>::const_iterator longest_word = v.end();
list<string>::const_iterator it;
for (it = v.begin(); it != v.end(); ++it) {
string paraula = *it;
if (paraula.size() > mida_max) {
mida_max = paraula.size();
longest_word = it;
}
}
return longest_word;
}

int main() {
string word;
list<string> text;
while (cin >> word) {
text.push_back(word);
}
auto i = longest(text);
if (i != text.end()) {
cout << *i << endl;
} else {
cout << "<empty sequence>" << endl;
}
return 0;
}
```

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

LlistaIOParInt.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

LlistaIOParInt.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

```cpp
#include "ParInt.hh"

ParInt::ParInt(){
p=0;s=0;
}

ParInt::ParInt(int a,int b){
p=a;s=b;
}

int ParInt::primer() const{
return p;
}

int ParInt::segon() const{
return s;
}

bool ParInt::llegir(){
cin >> p >> s;
return (p!=0 or s!=0);
}

void ParInt::escriure() const{
cout << p << ' ' << s << endl;
}
```

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

ParInt.hh

```
#ifndef CLASS_ParInt_HH
#define CLASS_ParInt_HH

#include <iostream>
using namespace std;

class ParInt {

private:

int p;
int s;

public:

//Constructores
ParInt();
/* Pre: cert */
/* Post: el resultat es el parint (0,0) */
ParInt(int a,int b);
/* Pre: cert */
/* Post: el resultat es el parint (a,b) */

//Consultores
int primer() const;
/* Pre: cert*/
/* Post: retorna el valor de p */
int segon() const;
/* Pre: cert*/
/* Post: retorna el valor de s */

//Entrada/sortida
bool llegir();
/* Pre: cert*/
/* Post: llegeix dos enters i els assigna al parametre implicit, a mes, */
/* si llegeix el parell 0 0 retorna fals, altrament retorna cert */

void escriure() const;
/* Pre: cert */
/* Post: escriu el parametre implicit per la sortida estandard */

};
#endif
```

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

program.cc

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

#include <list>

#include "ParInt.hh"
#include "LlistaIOParInt.hh"

ParInt BuscarLlista(list<ParInt> llista, int n) {
int repe = 0;
int suma = 0;
while (!llista.empty()) {
if (llista.front().primer() == n) {
++repe;
suma += llista.front().segon();
}
llista.pop_front();
}
return ParInt(repe, suma);
}


int main() {
list<ParInt> llista;
LlegirLlistaParInt(llista);
int n;
cin >> n;
ParInt resultats = BuscarLlista(llista, n);
cout << n << ' ' << resultats.primer() << ' ' << resultats.segon() << endl;
}
```

Exercici: X21215 Implementació de matrius esparses

PRO2Excepcio.hh

```
#ifndef PRO2_EXCEPCIO // per evitar fer mes d'un include d'aquest fitxer
#define PRO2_EXCEPCIO

#include <exception>

using namespace std;

/* Classe PRO2Excepcio */
class PRO2Excepcio: public exception {
public:
 PRO2Excepcio(const char* mot) : exception(), mensaje(mot) {}
 const char* what() const throw() {return mensaje;};
 private:
 const char* mensaje;
 };
 /* Fi classe PRO2Excepcio*/

 #endif
```

Exercici: X21215 Implementació de matrius esparses

matriu.hh

```
#include <iostream>
#include <list>
#include <vector>

#include "PRO2Excepcio.hh"

using namespace std;

template <typename T>
class matriu {
private:
struct parint {
int col;
T val;
};

vector<list<parint> > files; // elements de la matriu int nfil; // dimensions de la matriu int ncol;

// 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.
public:
// Constructores
matriu(int m, int n);
/* Pre: n>0 */
/* Post: matriu de mXn amb tots els elements a 0 */

// Modificadores
void modif_pos(int i, int j, T x);
/* Pre: 0<=i< num files del par■metre impl■cit; 0<=j< num columnes del par■metre impl■cit */
/* Post: el par■metre impl■cit queda com l'original per■ amb x a la la posici■ [i][j] */

void suma(const matriu& m1, const matriu& m2);
/* Pre: m1 i m2 tenen la mateixa dimensi■ */
/* Post: el par■metre impl■cit ■s la suma de m1 i m2 */

void producte(const matriu& m1, const matriu& m2);
/* Pre: el n■mero de columnes de m1 ■s igual al n■mero de files de m2 */
/* Post: el par■metre impl■cit ■s el producte de m1 i m2 */

// Consultores
int num_files() const;
/* Pre: cert */
/* Post: El resultat es el n■mero de files del par■metre impl■cit */

int num_columnes() const;
/* Pre: cert */
/* Post: El resultat es el n■mero de columnes del par■metre impl■cit */

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 valores 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

```
#include "matriu.tcc"

int main() {

int operacio;
cin >> operacio;

matriu <int> m1(0,0);
int nf1, nc1;
cin >> nf1 >> nc1;
m1.llegeix (nf1,nc1);

int nf2, nc2;
cin >> nf2 >> nc2;

matriu <int> m2(0,0);
m2.llegeix(nf2,nc2);

m1.escriu();
m2.escriu();

if (operacio == 1) {
matriu <int> s(0,0);

s.suma(m1,m2);
s.escriu();
}

if (operacio == 2) {
matriu <int> p(0,0);

p.producte(m1,m2);
p.escriu();
}

}
```

Exercici: 4.2 Llistes

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

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

void min_max(const list<int>& l, int& min, int& max) {
list<int>::const_iterator it = l.begin();
min = max = *it;
for (++it; it != l.end(); ++it) {
if (min > *it)
min = *it;
else if (max < *it)
max = *it;
}
}

void borra(list<int>& llista, float& suma, int n, int& size) {
for (list<int>::iterator it = llista.begin(); it != llista.end(); ++it) {
if (*it == n) {
llista.erase(it);
--size;
suma -= n;
return;
}
}
}

int main() {
list<int> llista;
int codi, n, min, max;
float suma = 0;
int size = 0;
cin >> codi >> n;
min = max = n;
while (n != 0 || codi != 0) {
if (codi == -1) {
if (size == 0) {
min = n;
max = n;
} else if (n < min)
min = n;
else if (n > max)
max = n;
llista.push_back(n);
++size;
suma += n;
} else if (!llista.empty() && codi == -2) {
borra(llista, suma, n, size);
if (!llista.empty() && (min == n || max == n)) min_max(llista, min, max);
}

if (llista.empty()) {
cout << 0 << endl;
suma = 0;

} else
```

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

Exercici: 4.2 Llistes

X60047 Punt mig d'una llista.cc

```cpp
using namespace std;
#include <iostream>
#include <list>
#include <string>

int main() {
list<int> llista;
list<int>::iterator mid = llista.end();
int x, size = 0;
string instruccio;

while (cin >> instruccio) {
if (instruccio == "get_mid_value") {
if (size % 2 == 0)
cout << "error" << endl;
else
cout << *mid << endl;
} else if (instruccio == "push_front") {
cin >> x;
llista.push_front(x);
++size;
if (size == 1) {
mid = llista.begin();
} else if (size % 2 == 1) {
--mid;
}
} else if (instruccio == "push_back") {
cin >> x;
llista.push_back(x);
++size;
if (size == 1) {
mid = llista.begin();
} else if (size % 2 == 0) {
++mid;
}
} else if (instruccio == "pop_front") {
if (llista.empty()) {
cout << "error" << endl;
} else {
if (size % 2 == 1) {
++mid;
}
llista.pop_front();
--size;
if (size == 0) {
mid = llista.end();
}
}
} else if (instruccio == "pop_back") {
if (llista.empty()) {
cout << "error" << endl;
} else {
if (size % 2 == 0) {
--mid;
}
```

```
56 llista.pop_back();
57 --size;
58 if (size == 0) {
59 mid = llista.end();
60 }
61 }
62 }
63 }
64 }
```

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,
uno.end());
20 }
```