

Guia de correcció dels exàmens de PRO1

Jorge Castro i Lluís Màrquez

Curs 2026–2027

Consideracions Generals

- Cada exercici de l'examen serà **qualificat exclusivament pel professor**. Per tant, desapareix l'anomenada nota automàtica o nota del Jutge. El resultat dels enviaments al Jutge tindrà caràcter informatiu i podrà ser tingut en compte pel professor durant la correcció.
- Aquest document estableix una llista de deficiències i errors generals que comporten penalitzacions en la qualificació d'un exercici. En general, corresponen a qüestions d'estil i de bones pràctiques de programació. S'organitzen en tres categories: **deficiències menors**, **deficiències majors** i **problemes d'eficiència**. En un mateix exercici, cada concepte es penalitzarà com a màxim una vegada. Les penalitzacions indicades assumeixen que els exercicis es puntuen sobre 10.
- L'enunciat d'un exercici pot establir requeriments o penalitzacions específics diferents dels recollits en aquest document. En cas de discrepància, **sempre prevaldran les indicacions de l'enunciat**.
- En general, qualsevol solució que incompleixi una restricció explícita de l'enunciat serà qualificada amb un zero. Per exemple, aquest seria el cas d'una solució iterativa quan l'enunciat exigeix explícitament una solució recursiva.
- Pel que fa als aspectes específics de cada problema —és a dir, fins a quin punt el programa resol correctament el problema plantejat i com s'han de penalitzar els errors funcionals— es proporcionaran criteris de correcció específics abans d'iniciar el procés de correcció. Aquests criteris seran públics en el moment que es publiquin les notes.

1 Deficiències Menors

Cada tipus de deficiència menor comporta una penalització de 0,5 punts i es penalitza com a màxim una vegada per exercici. La penalització total per deficiències menors no pot superar els 2 punts per exercici.

- **Codi redundant o innecessàriament complicat.** Exemples:

- `return suma + 0;` → ha de ser `return suma;`
- `return trobat == true;` → ha de ser `return trobat;`

```
if (trobat) return true;  
else return false;
```

- ha de ser `return trobat;`

- En general, qualsevol bloc de codi que sigui fàcilment simplificable.

- **Esquemes de lectura no recomanats.** Es penalitzaran els esquemes de lectura que compliquin innecessàriament les condicions d'aturada o que no aprofitin informació coneguda sobre la seqüència d'entrada. Per exemple:

- Esquema `while (cin >> k)` quan es coneix la mida de la seqüència d'entrada.

Esquema esperat quan es coneix la mida de la seqüència:

```
// n és la mida de la seqüència d'entrada
for (int i = 0; i < n; ++i) {
    cin >> k;
    ...
}
```

- Esquema `while (cin >> ch and ch != marca_fi)` quan la seqüència té una marca de fi (o centinella).

Esquema esperat:

```
cin >> ch;
while (ch != marca_fi) {
    ...
    cin >> ch;
}
```

- **Variables declarades en un àmbit més ampli del necessari**, quan aquesta pràctica es produeixi de manera sistemàtica (no atribuïble a un descuit).
- **Noms de variables que dificulten la lectura del codi.** Es penalitzarà l'ús generalitzat de noms arbitraris o poc informatius quan les variables tinguin una interpretació clara en el context del problema (per exemple, `hores`, `minuts`, `graus` o `nom`). També es penalitzaran els noms ofensius o que introdueixin confusió, com ara l'ús de negacions en variables booleanes:

```
bool no_trobat = true;
```

- **Formatació que dificulta la lectura del codi.** Ús sistemàtic de línies excessivament llargues o d'un espaiat irregular que provoqui codi excessivament apilat o dispers, etc.
- **Dues o més variables innecessàries.**
- **Utilitzar enters 0/1 com a booleans.**
- **Ús inadequat d'un for en lloc d'un while.** S'ha d'utilitzar un `while` quan la iteració està determinada per una condició i no segueix de manera natural un esquema controlat per un comptador. Per exemple:

```
for (int i = 0; x%2 == 0; ++i) x = x/2;
```

hauria de ser:

```
while (x%2 == 0) x = x/2;
```

2 Deficiències majors

Cada tipus de deficiència major comporta una penalització d'un punt i es penalitza com a màxim una vegada per exercici. No hi ha un límit global per a la penalització acumulada per deficiències majors.

- **Fer una lectura sense tenir la garantia que hi ha una dada disponible.** Per exemple, intentar llegir un element d'una seqüència quan aquesta pot haver finalitzat.
- **Consultar una variable que no ha estat inicialitzada.**
- **Modificar la variable de control d'un for dins del cos del bucle.**
- **Avaluar innecessàriament una condició costosa abans d'una de simple.** Aquest criteri només s'aplicarà en casos evidents en què l'ordre de les condicions permeti evitar, gràcies a l'avaluació en curtcircuit, un càlcul significativament més costós. Per exemple:

```
Matrix mat = ...
if (es_simetrica(mat) and mat[0][0] == 0) ...
```

ha de ser:

```
if (mat[0][0] == 0 and es_simetrica(mat)) ...
```

- **Indentació incorrecta de manera sistemàtica**, no atribuïble a un descuit puntual i que dificulti la lectura del programa.
- **No aplicar disseny descendent en problemes complexos**, donant lloc a blocs de codi innecessàriament llargs, difícils d'entendre, amb repeticions, i en general sense introduir funcions quan la descomposició és natural.
- **No documentar les funcions** mitjançant una especificació de tipus pre/post.
- **Fer crides repetides i innecessàries a una mateixa funció** quan el resultat es pot calcular una sola vegada i emmagatzemar en una variable per reutilitzar-lo posteriorment.
- A la capçalera d'una funció, **utilitzar inadequadament el pas de paràmetres per valor o per referència**. Per exemple, passar un vector per valor quan no hi ha cap raó que justifiqui la còpia (raó molt improbable en el nostre curs).
- **Utilitzar push_back() innecessàriament**. Per exemple quan la mida del vector es coneix prèviament i és més natural crear-lo directament amb la mida adequada.

```
// n és la mida del vector
vector<int> v;
for (int i = 0; i < n; ++i) {
    int x;
    cin >> x;
    v.push_back(x);
}
```

En lloc de:

```
vector<int> v(n);
for (int i = 0; i < n; ++i) cin >> v[i];
```

3 Ineficiències

En general, es consideraran errors greus i, per tant, comportaran penalitzacions més grans que les associades a les deficiències majors.

- Emmagatzemar una seqüència en un vector quan no és necessari per resoldre el problema.
- Implementar un recorregut en comptes d'una cerca.

- Utilitzar un algorisme amb una complexitat asimptòtica superior a la necessària. Per exemple, $O(n^2)$ en lloc de $O(n)$, o $O(n)$ en lloc de $O(\log n)$.

4 Altres

- En general, es desaconsella l'ús de construccions sintàctiques o funcions de les biblioteques de C++ que no s'hagin explicat a classe ni s'introdueixin explícitament a l'enunciat. El seu ús pot comportar penalitzacions en la qualificació de l'exercici.