PROGRAMACIÓ CIENTÍFICA. SEMESTRE DE TARDOR. PRÀCTIQUES DE LABORATORI D'ORDINADORS

LLISTA 2. PUNTERS

1 Feu una funció de tipus `void` i amb dos arguments, a la qual se li passen dos enters positius, i els canvia pel seu *màxim comú divisor* (mcd) i el seu *mínim comú múltiple* (mcm).

Completeu el programa amb una funció `main` on es llegeixen 2 enters, es comprova que són positius, es crida la funció anterior, i s'escriu el mcd i el mcm.

2 Feu un programa que llegeixi una línia de text (de 80 caràcters com a màxim i sense usar accents), i cridi una funció on *s'analitza la línia*, en el sentit que s'explica a continuació. Cal comptar quants caràcters hi ha de cadascun dels 5 tipus següents: vocals, consonants, dígits numèrics, espais en blanc, i altres caràcters. El programa principal escriurà després els resultats.

Per a llegir una línia, podeu usar la funció `gets()`, la qual dona errors de compilació (no en feu cas), o també la funció `scanf()`, amb els especificadors especials per a cadenes de caràcters (secció 4.1.2 dels apunts del departament).

Nota. Per a fer la classificació, us pot servir la consulta d'una taula de caràcters ASCII. Però no és imprescindible.

3 (inversió d'un enter) Feu un programa que llegeixi un enter no nul i escrigui l'enter que resulta de canviar l'ordre dels dígits.

Cal decidir què es fa amb el signe i els zeros. Ho exposem amb exemples: si llegeix -1234 llavors ha d'escriure -4321; si llegeix 56700 llavors ha d'escriure 00765; si llegeix 0567 llavors ha d'escriure 765.

En particular, heu de fer una funció de prototipus `int invertir (int *m);` que retorni el nombre de zeros en què acaba el valor de l'argument, i que inverteixi l'argument. Per a fer la inversió, cal separar els dígits un a un, guardant-los en un vector (no pot haver més de 10 dígits, en un `int`), i després cal construir el valor invertit.

4 (potència d'un polinomi) Feu una funció que calculi el producte de dos polinomis, $p(x)*q(x)$. El prototipus ha de ser

`int prodpol(int n, double *p, int m, double *q, double *pq);`

i ha de retornar el grau del producte.

Feu també una funció `main` on es llegeixen: el grau i els coeficients d'un polinomi $p(x)$, i un enter positiu k. Cal calcular el polinomi $p(x)^k$ i escriure els seus coeficients.

Per a guardar un polinomi, només cal guardar els seus coeficients en un vector. Cal usar memòria dinàmica.

5 (accés a elements de matrius sense usar [ ]) Feu un programa que llegeixi una matriu A, de dimensió $n \times n$, i un vector x, de n components, i calculi A^2x de dues maneres: $A(Ax)$ i $(AA)x$.

Podeu usar memòria dinàmica, o no. En el segon cas, podeu suposar que $n \leq N$, amb N conegut. Podeu declarar un vector auxiliar per a guardar (Ax) i una matriu auxiliar per a guardar (AA).

Però *NO podeu usar parèntesis quadrats* per a accedir a les components de les matrius i els vectors. Només els podeu usar en les declaracions.

6 Feu una funció per a calcular el *producte de dues matrius reals i simètriques* de dimensió n (atenció: la matriu resultant pot ser no simètrica).

La lectura de les dades i l'escriptura de les 3 matrius, es faran a la funció `main`. Heu d'usar memòria dinàmica.

Per a les dues matrius simètriques, només es reservarà el mínim espai necessari i, de fet, només es llegiran aquests elements (per a cadascuna, $n(n+1)/2$ elements).

7 (punters a funcions) Feu dues funcions en C que avaluïn dues funcions derivables de $[0,1]$ en $[0,1]$; per exemple, $f_1(x) = 3.99x(1-x)$ i $f_2(x) = 6.3x - 16x^2 + 10.7x^3$.

Feu una funció en C, de prototipus:

```
double g(double (*f)(double), double y, int n);
```

on s'avaluï $g(y) \equiv f^n(y) \equiv (f \circ \ldots^{(n)} \ldots \circ f)(y)$. O sigui, la funció g és la composició de n vegades una funció f, avaluada en y. La funció f és passa a la funció g com a primer argument, mitjançant un punter a funció.

Feu una funció *main* per a generar una taula de valors amb cinc columnes. A la primera hi haurà valors equidistant de y a l'interval $[0,1]$, separats entre si una determinada distància *pasy*. A les altres quatre columnes hi haurà els valors corresponents de les funcions $f_1^n(y)$, $(f_1^n)'(y)$, $f_2^n(y)$ i $(f_2^n)'(y)$.

Les derivades es calcularan només aproximadament, usant la fórmula

$$g'(y) \approx \frac{g(y+h) - g(y-h)}{2h} \ ,$$

on h és un valor positiu i molt pròxim a zero.

El programa ha de llegir els valors de *pasy*, n i de h. (per exemple, 0.01, 10 i 10^{-5}).

8 (funció de tipus punter, mètode de la potència per a calcular un vap/vep d'una matriu quadrada) En aquest exercici es demana d'usar una funció de tipus `double *`. Cal fer un `malloc` dins de la funció i el `free` corresponent en el `main`.

Feu un programa on:

- Es llegeix una dimensió n i els elements d'una matriu A, de dimensió $n \times n$. S'inicialitza un vector $x = (1, 1, \ldots, 1)$ de n components.
- Es va actualitzant el vector x de la manera següent: sigui $y = Ax$, es calcula la component de y que és màxima en valor absolut (sigui `ymax`), i sigui $x = \frac{1}{ymax} y$. El càlcul $y = Ax$ s'ha de fer en una funció de prototipus

```
double *Ax(int n, double **A, double *x);
```

Cal iterar les actualitzacions mentre: no se supera un nombre màxim d'iteracions permeses, i el valor `ymax` varia més que una determinada precisió desitjada.

En cada iteració, feu escriure el nombre d'iteració, el valor `ymax` i el vector `x`. Si hi ha convergència, llavors `ymax` és un valor propi de la matriu `A` i `x` és un vector propi associat.