PROGRAMACIÓ CIENTÍFICA. CURS 2019-20. PRIMAVERA. PRÀCTIQUES DE LABORATORI D'ORDINADORS

LLISTA 2. PUNTERS I MEMÒRIA DINÀMICA

1 (màxim comú divisor i mínim comú múltiple) 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 (classificació dels caràcters d'una cadena en diferents categories) Feu un programa que:

- Llegeix una línia de text (entrada per teclat, acabant prement la tecla ENTER).

- Crida una funció adequada, en la qual: es compta quants caràcters s'han llegit de cadascun dels cinc tipus següents: *vocals*, *consonants*, *dígits numèrics*, *espais en blanc* i *altres caràcters*; i retorna el nombre total de caràcters.

- Escriu quants caràcters hi ha de cada classe i el nombre total.

Notes:

i) No entreu caràcters accentuats perquè porten problemes.

ii) Fixeu, amb un `define`, el nombre màxim de caràcters (`N`) que pot tenir la línia.

iii) No llegiu mitjançant `gets`, perquè no controla la llargada de la lectura i, per tant, és un risc potencial de seguretat. Llegiu mitjançant la instrucció

```
fgets (linia, N+1, stdin);
```

on `stdin` indica l'entrada estàndard (teclat), `linia` és el nom de la cadena que s'omple amb la lectura, i `N+1` indica que, com a màxim, es llegiran `N caràcters` (cal guardar una posició per al caràcter de final de cadena).

iv) A diferència de `gets`, la funció `fgets` també posa a la cadena el caràcter de final de línia (suposant que no s'hagi acabat l'espai reservat per a la cadena). Tingueu present que aquest no s'ha de comptar.

3 (inversió d'enters) Feu un programa que vagi llegint valors enters, i els vagi escrivint amb l'ordre dels dígits al revés. El programa s'aturarà quan llegeixi el valor 0.

En particular, heu de fer una funció de prototipus

```
void invertir (int *m);
```

que inverteixi els dígits de l'argument. Concretament, cal fer: eliminar els zeros del final, separar els dígits que queden, guardant-los en un vector, i reconstruir el valor corresponent als dígits en ordre invers.

No cal escriure el signe + en els valors positius. Exemple d'execució:

```
nombre llegit   = 12345   nombre invertit = 54321
nombre llegit   = 102030400   nombre invertit = 4030201
nombre llegit   = -10305   nombre invertit = -50301
nombre llegit   = -159000   nombre invertit = -951
s'ha llegit el valor 0
```

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

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

i que retorni 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; es calcula el polinomi $p(x)^k$; i s'escriu el seu grau i 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 (memòria dinàmica per a matrius hessenberg) Una matriu $A = (a_{ij})$ quadrada s'anomena *Hessenberg superior* (HS) quan es verifica $a_{ij} = 0$, per a tots els possibles subíndexs i, j tals que $i > j + 1$. Aquests elements seran anomenats *no essencials*.

Feu una funció per a calcular el producte de dues matrius HS de la mateixa dimensió $n \geq 3$.

A la funció `main`, feu la lectura de les dades (la dimensió i els elements de les matrius, sense els elements 0 no essencials), després crideu la funció per al càlcul de la matriu producte, i finalment escriviu les 3 matrius senceres (també els elements 0 no essencials).

Heu d'usar memòria dinàmica, però sense reservar espai per als elements no essencials.

Nota. El producte de dues matrius HS no és una matriu HS, però "quasi"·

7 (derivació numèrica, 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 x, int n);
```

on s'avaluï $g(x) \equiv f^n(x) \equiv (f \circ \ldots^{(n)} \ldots \circ f)(x)$. O sigui, la funció g és la composició de n vegades la funció f, avaluada en x. 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 x a l'interval $[0, 1]$, separats entre si una determinada distància $pasx$. A les altres quatre columnes hi haurà les corresponents imatges de cada x per les funcions $f_1^n(x)$, $(f_1^n)'(x)$, $f_2^n(x)$ i $(f_2^n)'(x)$.

Les derivades es calcularan només aproximadament, usant la fórmula

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

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

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

Redirigiu la sortida estàndard a un fitxer i useu `gnuplot` per a pintar les 4 funcions calculades.

8 (funció de tipus punter, mètode de la potència per a calcular un vap/vep d'una matriu) 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:

(i) Es llegeix una dimensió n, i els elements d'una matriu A, de dimensió $n \times n$; i s'inicialitza un vector $x = (1, 1, \ldots, 1)$ de n components.

(ii) Es va actualitzant el vector x de la manera següent: es calcula el vector $y = Ax$, es calcula la component de y que és màxima en valor absolut (sigui `ymax`), i el nou vector x és defineix com $x = \frac{1}{ymax} y$. Programeu el càlcul del vector y en una funció de prototipus

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

Cal anar iterant l'actualització mentre no se superi un nombre màxim d'iteracions permeses, i mentre el valor `ymax` variï 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.