Llista 4: Funcions

59. Feu una funció, de nom `longitud`, que calculi la distància entre dos punts donats del pla. Useu-la en un programa que calculi el perímetre i l'àrea d'un triangle, donat per tres punts del pla (**triangle.c**).

60. a) Escriviu una funció de capçalera `int multi(int n, int p)` que retorni la quantitat de vegades que `n` és divisible per `p`.

    Per exemple, quan es crida la funció amb els valors $n = 63$ i $p = 3$, el valor a retornar ha de ser 2, ja que $63 = 3 \times 3 \times 7$.

    Nota: podeu suposar $n > 0$ i $p > 1$.

    b) Escriviu una funció `main` que llegeixi un nombre enter `num`, comprovi que el valor llegit és major que 1 (en cas contrari, avisi que la dada és incorrecta i acabi) i, fent ús de la funció de l'apartat anterior, escrigui la descomposició de `num` en els seus factors **primers**, seguint l'exemple. Si llegís el valor 126 hauria d'escriure:

```
Pel valor 126 tenim:
   divisor primer: 2, potència: 1
   divisor primer: 3, potència: 2
   divisor primer: 7, potència: 1
```

    (**descomposicio.c**)

61. Feu un programa que calculi un zero d'una funció f, amb tolerància `tol`, mitjançant el mètode de Newton-Raphson partint del valor x_0. Caldrà llegir el valor inicial x_0, la tolerància `tol` i el nombre màxim d'iterats, `iterMax`. (**newton.c**)

    El mètode de Newton (o de Newton-Raphson) permet calcular un zero d'una funció derivable:

    > Donada una aproximació inicial x_0, calculem la successió
    >
    > $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \ldots$$

    Per concloure que hi ha convergència, cal $|x_{n+1} - x_n| <$ `tol` o $|f(x_{n+1})| <$ `tol`. Com que el mètode pot no convergir, cal controlar el nombre màxim d'iteracions a fer (`iterMax`).

    Necessiteu crear i cridar dues funcions `double fun(double);` i `double dfun(double);` que retornen, respectivament, el valor de la funció f i de la seva derivada en un punt.

    Useu el programa per trobar els zeros de les funcions $f_1(x) = x^3 - 10\sin x$ i $f_2(x) = x^6 - x - 1$, amb tolerància 10^{-8}. Per a cada una d'aquestes funcions, escriviu en un fitxer diferent les funcions `fun` i `dfun` corresponents.

62. Donats dos vectors $x, y \in \mathbb{R}^n$, volem determinar l'angle que formen. Sabem que

$$\cos \widehat{xy} = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

on $x \cdot y$ és el producte escalar del vector x pel vector y i $\|x\|$ i $\|y\|$ són la norma de x i la de y, respectivament. Per definició $\|x\| = \sqrt{x \cdot x}$.

Escriviu un programa que llegeixi la dimensió dels vectors, les seves components i calculi l'angle que formen. Per calcular el producte escalar useu una funció `prodEscalar`. (**angleVectors.c**)

63. Dissenyeu una funció de nom `horner` que avalui un polinomi de grau n en un punt y usant la regla de Horner. Utilitzeu-la per veure si un polinomi pot tenir un zero entre dos valors a i b (**zero.c**).

Nota: Si $p_4(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ és un polinomi de grau 4, el seu valor en y es pot calcular com (començant els càlculs pels parèntesis més interiors)

$$p_4(y) = ((((a_4)y + a_3)y + a_2)y + a_1)y + a_0$$

64. Feu un programa que, donats dos vectors `primeraAssig` i `segonaAssig` que contenen les notes dels n alumnes de dues assignatures, escrigui quina de les dues assignatures té la mitjana de les qualificacions dels alumnes presentats més gran. Si un alumne no s'ha presentat a una assignatura, constarà un -1 com a qualificació.

La nota mitjana es calcula mitjançant una funció `mitjana`, que rep el vector v que conté les notes i el nombre d'alumnes matriculats n, i la retorna. (**mitjanaMajor.c**)

65. En un estudi ambiental s'han mesurat les temperatures i les pressions atmosfèriques de diferents poblacions durant n dies, i volem veure si entre aquests dos valors hi ha correlació lineal. Per a veure això calcularem la recta de regressió de la temperatura (y) sobre la pressió (x).

$$y - \bar{y} = \frac{\sigma_{xy}}{\sigma_x^2}(x - \bar{x})$$

Per a determinar la recta de regressió necessitarem la mitjana de cada variable ($\bar{x}$ i $\bar{y}$), la variança de x ($\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$) i la covariança de x i y ($\sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$).

a) Feu una funció **`double`** `mitjana(`**`double`** `v[], `**`int`** `n)` que calculi la mitjana aritmètica de les n dades del vector v i retorni el valor de la mitjana.

b) Feu una funció **`double`** `varianca(`**`double`** `v[], `**`int`** `n)` que calculi la variança de les dades del vector v i retorni aquest valor.

c) Feu una funció **`double`** `covarianca(`**`double`** `v[], `**`double`** `w[], `**`int`** `n)` que calculi la covariança de les dades dels vectors v i w i retorni aquest valor.

d) Feu una funció principal que llegeixi les dades de 30 dies corresponents a una ciutat, guardant la temperatura en un vector i la pressió atmosfèrica en un altre, i escrigui la recta de regressió. El programa demanarà si es vol calcular la recta d'una altra població o no; s'haurà d'entrar un 1 per continuar i un 2 per aturar-se. (**rectaRegressio.c**)

Les dades es donaran dia a dia de la forma:

11.8 1021.4
13.2 1019.3
...

66. Volem calcular $y = ABx$, on A és una matriu real de dimensió $m \times n$, B una matriu real de dimensió $n \times m$, i x i y dos vectors de m components reals, i després determinar si y és múltiple de x.

    Feu una funció de nom `prodMatVect`, que calculi el producte d'una matriu per un vector.

    Feu una funció `main`, que llegeixi dos enters, m i n, els elements de A per files, els elements de B per files, els elements de x, i usant la funció `prodMatVect`, calculi $y = ABx$, i determini si y és múltiple de x. No calculeu el producte AB.

    Caldrà escriure A, B, x i y encolumnadament, obtenir els mòduls de x i y i escriure'ls i, finalment, determinar si y és múltiple de x, donant el missatge corresponent. (**productesMatVect.c**)

67. Feu una funció `void matriuRotacio(double rot[][3], double alfa, int eix)` que ompli una matriu de rotació de dimensió 3 d'angle α al voltant de l'eix indicat pel tercer paràmetre (1: x, 2: y, 3: z). En el cas de l'eix de les x la matriu de rotació té la forma

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

    Alguns moviments efectuats per un robot són el resultat d'aplicar diverses rotacions. Escriviu un programa que llegeixi tres angles β, ϕ i θ i trobi la matriu resultant d'aplicar una rotació al voltant de l'eix de les x d'angle β, després una rotació al voltant de l'eix de les y d'angle ϕ i finalment una rotació al voltant de l'eix de les z d'angle θ. (**rotacions.c**)

    Nota: recordeu que el producte de matrius no és commutatiu.

68. Feu una funció `esVocal`, que, donat un caràcter, retorna un 1 si és una vocal i un 0 si no ho és. Definiu a la funció un vector `vocal[11] = "aeiouAEIOU"` per comprovar-lo.

    Escriviu una funció `main` que llegeixi una frase i, mitjançant la funció anterior, suprimeixi totes les vocals de la frase, guardant-la en la mateixa, i després escrigui la frase sense vocals. (**suprimirVocals.c**)

69. Escriviu una funció de nom `esDeLaTira`, que té per arguments un caràcter i una tira i retorna 1 si el caràcter es troba a la tira i 0 altrament.

Escriviu un programa que llegeixi un text d'un màxim de 100 caràcters i l'escrigui posant cada paraula en una línia, suprimint els signes de puntuació. Usarà la funció `esDeLaTira` amb `".:,;()[]-{} "`. (**paraulaPerLinia.c**)

70. a) Feu una funció **`int`** `divisors(`**`int`** `n, `**`int`** `div[31])`, a la qual se li passa el valor de n (enter entre 2 i 1000), omple les primeres components del vector `div` amb tots els divisors de n (exclós n), i retorna quants n'hi ha: q. El que hi hagi a la resta de components de `div` no ens importa.

Exemple. Si fos $n = 168$, llavors ompliria

$$\texttt{div} = (1,2,3,4,6,7,8,12,14,21,24,28,42,56,84,...)$$

i retornaria $q = 15$.

Nota. Se sap que, per a $n \leq 1000$, la quantitat de divisors de n no passa de 31.

b) Feu una funció **`int`** `base2(`**`int`** `m, `**`int`** `bits[32])`, a la qual se li passa el valor de m (enter més gran que 1), omple les primeres components del vector `bits` amb els dígits de l'expressió de m en base 2, i retorna la quantitat p de dígits calculats (fins a l'últim 1).

Exemple. Si fos $m = 14 = (1110)_2$ llavors seria `bits`$= (0,1,1,1,0,...,0)$ i retornaria $p = 4$.

c) Feu una funció `main` que llegeixi un valor enter n, comprovi que està situat entre 2 i 1000, i, usant les dues funcions anteriors, escrigui si n és, o no, semiperfecte.

Definició. Un nombre enter més gran que 1 s'anomena *semiperfecte* quan és igual a la suma d'algun subconjunt dels seus divisors (exclós ell).

Exemples

- El nombre $n = 20$ té divisors $1,2,4,5,10$ i es verifica $20 = 1+4+5+10$. Per tant, 20 és semiperfecte.
- El nombre $n = 15$ té divisors $1,3,5$. Les possibles sumes donen $0,1,3,5,4,6,8,9$. Cap d'elles no és 15. Per tant, 15 no és semiperfecte.

Nota. Com es poden generar tots els subconjunts d'un conjunt donat? Cal saber que, si un conjunt C té q elements, llavors hi ha 2^q subconjunts possibles de C (comptant el buit i el total).

Cada subconjunt es pot fer correspondre a un nombre entre 0 i $2^q - 1$. Escrivint aquest nombre en base 2, els uns en diuen quins elements de C s'han d'incloure en el subconjunt.

Exemple. Sigui $C = \{c_0, c_1, c_2, c_3, c_4\}$ un conjunt de $q = 5$ elements. S'han de generar $2^5 = 32$ subconjunts, cadascun associat a un valor m entre 0 i 31.
$m = 0 = (00000)_2$ correspon al subconjunt buit.
$m = 7 = (00111)_2$ correspon al subconjunt $\{c_0, c_1, c_2\}$.
$m = 18 = (10010)_2$ correspon al subconjunt $\{c_1, c_4\}$.
$m = 21 = (10101)_2$ correspon al subconjunt $\{c_0, c_2, c_4\}$.
$m = 31 = (11111)_2$ correspon al subconjunt total.

71. Escriviu una funció de nom `prod_mat` tal que, donades dues matrius C i D (C de $m \times n$, D de $n \times p$), calculi CD.

    Useu-la en un programa que llegeixi dues matrius A i B (A de $m \times n$, B de $n \times p$) i avalui AB, BA i $C = AB - BA$, sempre que sigui possible. Si alguna operació no és possible, doneu el missatge corresponent (**matrius.c**).

72. Feu un programa que llegeixi una frase i digui si és o no un palíndrom. (**palindrom.c**)

    Un palíndrom és una frase que es llegueix igual d'esquerra a dreta que de dreta a esquerra. Exemple: "Dábale arroz a la zorra el abad".

    La funció `main` llegeix una frase i escriu si la frase és o no un palíndrom. Podeu suposar que la frase s'introdueix sense accents i en minúscules.

    Feu dues funcions `invertir`, que rep una tira i la inverteix sobre ella mateixa, i una funció `senseBlancs`, que rep una tira i n'elimina els espais en blanc.

73. Feu una funció de nom `identiques` que, donades dues paraules, retorni un 1 si són iguals, i un 0, altrament.

    Escriviu un programa que llegeixi una frase i, usant la funció anterior, compti les vegades que apareix la primera paraula. (**paraulesIdentiques.c**)

74. Escriviu dues funcions per calcular la suma de dos polinomis i una altra per calcular el producte de dos polinomis.

    ```
    int suma(double p[], int grP, double q[], int grQ, double polSuma[])
    ```

    ```
    int producte(double p[], int grP, double q[], int grQ, double polProd[])
    ```

    que, donats dos polinomis `p` (de grau `grP`) i `q` (de grau `grQ`), calculen el polinomi suma `polSuma` i el polinomi producte `polProd`, respectivament, retornant el grau del polinomi resultant.

    Useu-les en un programa que llegeixi grau i coeficients de tres polinomis i calculi el polinomi suma dels tres, el polinomi producte del primer pel segon i el polinomi producte del primer pel tercer. (**sumaproductePolin.c**)