Matemàtica computacional i analítica de dades
Algorítmia i combinatòria en grafs...
Curs 2020–21

1 Bàsics de programació en C

1.1 Un exemple "simple"

El codi (Python) del requadre que ve a continuació defineix una funció que, quan s'executa, demana els tres coeficients d'una equació de segon grau i fa els càlculs necessaris per a donar com a resultat les seves solucions (sense resultat si les solucions són complexes)

```
1  def eqseggrau():
2      from numpy import sqrt
3      tol=1.e-10
4      a=float(input("a? "))
5      b=float(input("b? "))
6      c=float(input("c? "))
7      if abs(a)<tol:
8          print("Equacio de grau 1. (a es 0)")
9          return -c/b
10     discr=b**2-4*a*c
11     da=2*a
12     if discr<0.:
13         print("Discriminant negatiu. (Solucions complexes)")
14         return
15     if discr<tol:
16         print("Discriminant nul. (Solucio real doble)")
17         return -b/da
18     discr=sqrt(discr)
19     s=[(-b+discr)/da,(-b-discr)/da]
20     print("Les dues solucions de l'equacio ")
21     print("(", a, ")*x**2+(", b,")*x+(",c,")=0  son:\n",s[0],"\n",s[1])
22     return s
```

Podeu analitzar el seu funcionament executant (en una sessió de Python)

```
eqseggrau()
```

introduint les dades d'algunes equacions concretes. Per exemple:

- $x^2 + 2\,x + 1 = 0$ (solució doble $x = -1$).
- $2\,x^2 + x + 2 = 0$ (sense solucions reals).
- $3\,x^2 + 4\,x - 5 = 0$ (dues solucions diferents).

Noteu especialment que

- Com que s'han de calcular arrels quadrades cal disposar de la funció `sqrt( )`
- S'interpreten les dades que provenen de l'entrada com a valors `float` forçant el tipus.
- S'introdueix el llindar `tol` (10^{-10}) per sota del qual s'entén que un valor (positiu) és indistingible de 0. (Quan tenim quantitats representades en *punt flotant* la igualtat a 0 **mai** es produeix).

- Les instruccions `if` no tenen associades cap `else` ja que, en cas de verificar-se les condicions corresponents, el programa acaba en la instrucció `return` i la resta de codi ja no s'executa.
- La funció produeix dos tipus de resultat: *l'explicatiu* amb els missatges que generen les instruccions `print` i el *valor de sortida de la funció* que és el que es genera amb els `return`

El codi que apareix a continuació és una proposta en C per tal de fer (gairebé) el mateix. Després d'escriure el codi en un fitxer i compilar-lo[1] obtindreu un programa que realitza les mateixes operacions que la funció anterior.

```
/*
 Programa per a solucionar equacions polinomiques de segon grau.
*/
#include <stdio.h>
#include <math.h>
#define tol 0.000000001
int main(){
    double a,b,c,discr;
    printf("\nIntroduiu les dades a,b,c del polinomi a*x^2+b*x+c:\n");
    while(scanf("%lf %lf %lf",&a,&b,&c)<3){
        while (getchar()!='\n'){}
        printf("Les dades no han entrat be....\nTorneu-hi...\n");
    }
    if(fabs(a)<tol){
        printf("L'equacio no es de segon grau si poseu a=0 .\n");
        printf("La solucio de\n (%g) * x + (%g) =0\nes:\n",b,c);
        printf("x= %g\n",-c/b);
    }
    else{
        discr= b*b -4*a*c;
        if(discr<0){
            printf("\nEquacio sense solucions reals\n");
            printf("(%g)*x^2+(%g)*x+(%g)=0\n",a,b,c);
        }
        else{
            if(discr<tol){
                printf("\nEquacio amb solucio doble:\n");
                printf("La solucio de l'equacio\n");
                printf("(%g)*x^2+(%g)*x+(%g)=0\n",a,b,c);
                printf("es\n");
                printf("x=%g\n",-b/(2*a));
            }
            else{
                discr=sqrt(discr);
                printf("\nEquacio amb dues solucions.\n");
                printf("Les solucions de l'equacio\n");
                printf("(%g)*x^2+(%g)*x+(%g)=0\n",a,b,c);
                printf("son\n");
                printf("x=%g, x=%g\n\n",(-b+discr)/(2*a), (-b-discr)/(2*a));
            }
        }
    }
    return 0;
}
```

[1] Per exemple amb `gcc -Wall -o arxiu.exe arxiu.c -lm` dins d'una finestra de terminal o des del vostre entorn de programació favorit.

Noteu que en aquest cas s'han introduït (intencionadament) petites diferències en el flux de treball. En podeu detectar algunes? Quina finalitat es busca amb el codi de les línies 10–13?

1.2 Exercicis

Afegiu com a primera línia del programa i en format comentari els vostres `Nom`, `Cognom` i `NIU`.

El nom dels programes que contenen el codi ha de ser de la forma `PrXExY.c`, on `X` fa referència a la pràctica i `Y` a l'exercici. Per exemple, el nom del programa de l'apartat següent hauria de ser `Pr1Ex121.c`.

Exercici 1.2.1: Fixeu-vos que, com abans, el programa decideix que l'equació és de grau 1 sempre que el coeficient de x^2 que s'introdueixi tingui el valor absolut per sota d'una certa tolerància que es fixa en el símbol `tol` (així s'eviten les divisions per `a` quan aquesta variable conté un valor massa petit). Tot i aquesta precaució, podria ser que el valor que s'introdueix en la variable `b` també fos assimilable a 0 i, en aquest cas, el programa també hauria de fer una divisió per 0.

Feu una modificació del programa per tal d'evitar aquesta altra possibilitat de divisió per 0 fent que aparegui un missatge informant que s'han introduït dades *que generen una equació sense incògnita* i aturant en aquest punt el programa.

Exercici 1.2.2: Haureu notat, també, que el programa de l'exemple anterior acaba sense donar cap resultat en el cas que l'equació no tingui solucions reals. Feu una modificació al programa per tal que en aquest cas doni el valor de les solucions complexes de la forma $((-b \pm i\sqrt{4\,a\,c - b^2})/(2\,a))$.

Exercici 1.2.3: Finalment, tot i que les variables `a, b, c...` es declaren de tipus `double` els resultats es presenten en format genèric `%g` i es perd precissió en els resultats que es visualitzen. Feu que tots els resultats apareguin en un format que tingui, com a mínim, 12 xifres decimals.

2 Estructures. Assignació dinàmica de memòria

Si us connecteu a la pàgina web del IDESCAT

`http://www.idescat.cat/pub/?id=aec&n=214`

veureu les dades meteorològiques més significatives de totes les comarques de Catalunya de l'any 2018 (que són les dades més recents que hi ha publicades en aquest moment). Aquestes dades apareixen en dues taules separades i per a cada una de les files d'aquestes taules apareix al principi la mateixa informació (comarca, nom de l'estació d'on s'obtenen les dades i altitud de l'estació sobre el nivell del mar). Les columnes que queden de la primera taula informen de la temperatura mitjana, la mitjana de les temperatures màximes, la mitjana de les temperatures mínimes, la temperatura màxima i la temperatura mínima detectades durant l'any. En la segona taula s'agrupa la precipitació, humitat relativa, velocitat i direcció predominant del vent.

Per a fer la feina d'aquesta pràctica haureu de descarregar el fitxer, que trobareu al CV, `MeteoCat2018.txt`. Dins aquest fitxer hi trobareu les dades meteorològiques organitzades de forma que els camps d'informació queden separats per punts i comes (`;`). (Notareu que no hi ha accents, però teniu en compte que fer que aquests caràcters es llegeixin bé segons la codificació i es puguin transportar d'un sistema operatiu a un altre pot resultar una mica feixuc i, en alguns casos, complicat).

El programa que hi ha a continuació llegeix aquestes dades i, un cop fet això, presenta en pantalla el llistat de totes les comarques, l'estació d'on s'obtenen les dades i la temperatura màxima registrada.

Fixeu-vos que:

- En la línies 5–11 es defineix una tipus de dada `struct`, que es designa com `MetCom`, i permet tenir les dades d'una comarca agrupades en un sol bloc. Dins aquest `struct` apareixen dos camps de tipus `char` per tal de guardar els noms de comarca i les estacions respectives, un camp de tipus `int` per l'alçada sobre el nivell del mar, camps `float` per a les dades de temperatura, pluviometria, humitat relativa i velocitat del vent i, finalment, un altre camp `char` que contindrà la direcció del vent.

- El `while` de les línies 23–25 llegeix, caràcter a caràcter, el contingut del fitxer de dades i incrementa el comptador `numcom` cada cop que hi ha un salt de línia. Això fa que al final de tot el recorregut, i tenint en compte com estan estructurades les dades, el valor contingut en aquesta variable sigui el nombre de comarques de la llista.

- Un cop es coneix el nombre de comarques de la llista es reserva en la línia 30 un bloc de memòria per al punter `comarca`, de la mida adequada per tal que es pugui fer servir per guardar tota la informació de les comarques en blocs del tipus `MetCom`. Noteu que la variable `comarca` funcionarà com un vector de mida `numcom` amb dades de tipus `MetCom`.

- El que queda de programa s'encarrega de llegir les dades i, al final, escriure el resum.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct{
6      char nom[20];
7      char estacio[35];
8      int alt;
9      float tmtj,tmxm,tmnm,tmx,tmn,pluja,hr,vv;
10     char ddv[3];
11 }MetCom;
12
13 int main(){
14     FILE * meteo;
15     MetCom *comarca;
16     unsigned i=0,numcom=0,ll;
17
18     if ((meteo=fopen("MeteoCat2018.txt","r"))==NULL){
19         printf("No es pot obrir el fitxer\n");
20         return 1;
21     }
22
23     while((ll=fgetc(meteo)) != EOF){
24         if (ll=='\n'){numcom++;}
25     }
26     printf("\nDades de %d comarques\n",numcom);
27
28     rewind(meteo);
29
30     if((comarca = (MetCom *) malloc(numcom * sizeof(MetCom))) == NULL){
31         printf ("\nNo es possible assignar la memoria necessaria...\n\n");
32         return 1;
```

```
33      }
34
35      for(i=0;i<numcom;i++){
36          fscanf(meteo, "%[a-zA-Z'. -];", comarca[i].nom);
37          fscanf(meteo, "%[a-zA-Z'. -];", comarca[i].estacio);
38          fscanf(meteo, "%i;", &comarca[i].alt);
39          fscanf(meteo, "%f;", &(comarca[i].tmtj));
40          fscanf(meteo, "%f;", &(comarca[i].tmxm));
41          fscanf(meteo, "%f;", &(comarca[i].tmnm));
42          fscanf(meteo, "%f;", &(comarca[i].tmx));
43          fscanf(meteo, "%f;", &(comarca[i].tmn));
44          fscanf(meteo, "%f;", &(comarca[i].pluja));
45          fscanf(meteo, "%f;", &(comarca[i].hr));
46          fscanf(meteo, "%f;", &(comarca[i].vv));
47          fscanf(meteo, "%s\n", comarca[i].ddv);
48      }
49      printf("Fi de la lectura.....\n\n");
50
51      fclose(meteo);
52
53      printf("   Comarca              Estacio                                  Max\n");
54      printf("=================================================================\n");
55      for(i=0;i<numcom;i++){
56          printf("%2u ",i+1);
57          printf("%-20s ",comarca[i].nom);
58          printf("%-35s ",comarca[i].estacio);
59          printf("%-.1f",comarca[i].tmx);
60          printf("\n");
61      }
62
63      return 0;
64  }
```

2.1 Exercicis

Exercici 2.1.1: Afegiu un bloc en el que es calculin les mitjanes de la sèrie de les temperatures màximes registrades, la de les temperatures mínimes registrades i la de la pluviositat total de les comarques. Un cop fets els càlculs feu que apareguin els resultats a la pantalla.

Feu que també es localitzin i s'ensenyin a la pantalla les comarques amb la temperatura màxima més alta registrada i la de la temperatura mínima més baixa registrada.

Suggeriments: Per a fer aquest exercici es pot triar, clarament, entre dues estratègies que són complementàries i alternatives:

(a) Fer una abstracció del problema (programació estructurada) i definir una funció del tipus

```
float mitjana(float llista[], int nelem)
```

en la que l'entrada consisteixi en un vector `llista`, que conté els valors, i un nombre enter `nelem`, que sigui el nombre d'elements de la llista, i el resultat la mitjana corresponent. De forma anàloga fer una funció del tipus

```
void maxmin(float llista[], int nelem, float *vmx, float *vmin,
                        int *pmx, int *pmin)
```

on, a part de les dades anteriors, els punters `vmx, vmin, pmx, pmin` serveixin per assignar a les variables que es vulgui els valors màxim i mínim de la llista i, a més, les posicions en la llista on aquests valors màxim i mínim s'assoleixen. Noteu que, en aquesta opció, s'hauran de guardar les dades en un vector auxiliar. Això planteja dues opcions: fer tres passades sobre la llista amb totes les dades o bé passar un sol cop sobre la llista però introduir tres vectors auxiliars.

(b) Generar un únic *bucle* que vagi passant per tots els elements de la llista de dades (un sol cop) i que, per a cada registre, es vagin acumulant les sumes de `tmx`, `tmn`, `pluja`, mantenint els màxims i els mínims de les variables que corresponguin per tal d'obtenir, al final, els resultats que es busquen.

Exercici 2.1.2: (Fi de festa) Si us fixeu, les dades del nom de la comarca i de l'estació meteorològica corresponent es guarden en camps de l'estructura `MetCom` de mida fixa (20 i 35 caràcters) mentre que hi ha algunes d'aquestes dades que poden ser molt curtes. Una forma d'estalviar espai de memòria consisteix a fer que els camps de l'estructura `MetCom`, en comptes de contenir els valors corresponents de cada comarca, siguin el valor d'un punter per al que es podrà reservar el bloc de memòria just per a contenir les dades. Això vol dir que hauríeu de modificar els programes anteriors per tal que funcionessin amb una estructura `MetComP` del tipus

```
typedef struct
{
    char *nom;
    char *estacio;
    int alt;
    float tmtj,tmxm,tmnm,tmx,tmn,pluja,hr,vv;
    char ddv[3];
}MetComP;
```

i en el moment de llegir les dades crear un punter i reservar-li l'espai necessari per a contenir el que s'ha llegit, guardant en l'estructura només aquesta adreça de memòria.

Instruccions finals

Quan acabeu la pràctica, feu el lliurament dels fitxers de codi (que tenen els noms de la forma `PrXExY.c` segons el que hem indicat anteriorment) a través del Campus Virtual des de l'apartat de lliuraments de l'assignatura.