Problemes capítol 5

Realitzar cada exercici donant el pseudocodi i/o el diagrama de flux i el codi C.

Considerar, en cada cas, els procediments/funcions que simplifiquen la solució del problema i la fan més intel·ligible.

Com a exemples més simples, pràcticament tots els problemes realitzats fins ara en capítols anteriors es poden desglossar en problemes per a aquest capítol amb diferents procediments/funcions. Per exemple, hi pot haver els procediments d'entrada de dades, càlcul i sortida de dades.

5.1 Realitzar un programa que calculi la sèrie de Fibonacci. Creeu un programa principal que cridi a una funció que retornarà cada nou terme de la sèrie a calcular. La crida passarà els termes necessaris per a realitzar el càlcul. Utilitzeu el pas de paràmetres per referència o per adreça.

Nota: Recordar la sèrie de Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

5.2 Implementar un algorisme que calculi la velocitat i l'alçada dels bots d'una pilota deixada anar des d'una alçada h. Se sap que la velocitat a la que arriba a terra la pilota és de $v = \sqrt{2 \cdot g \cdot h}$, gt= 9.8m/s^2, i l'alçada del bot següent a la que arriba és h'=3·h/5. Realitzar una taula (velocitat, alçada) de tots els bots que se succeeixen mentre els bots siguin superiors a un 1/9 del bot inicial. Implementar els càlculs en una funció.

Compareu el resultat amb el què passaria en cas que estiguéssiu sobre la superfície llunar, on gl= 1.7m/s^2.

5.3 Donats a, b, c, d, e i f, estudiar el sistema de dues equacions amb dues incògnites de la forma i donar el sesultat quan les rectes siguin secants.

$$a\,x + by = e$$
$$cx + dy = f$$

5.4 Programa que visualitza per pantalla un 1 entre 7 zeros que va desplaçant-se enmig d'ells (només un sentit). Segueix la seqüència (cas n=6):

100000-> 010000->....->000001->100000-> 010000->....

Donada la rapidesa de càlcul, s'ha de preveure una funció d'espera per a la visualització correcta.

5.5 Programa més general que en el cas anterior. Visualitzar per pantalla un 1 que va desplaçant-se enmig de n zeros, a un costat i a l'altre. Segueix la seqüència (exemple per n=6):

100000-> 010000->....->000001->000010->000100->...100000-> 010000->....

Característiques:

Permetre entrar el nombre de bits a visualitzar.
La posició del 1 es calcularà a partir d'una variable (no cal emprar 'case').
Fer el bucle d'espera emprant funcions temporals que proporciona la llibreria time.h.
Generar un 'beep' cada cop que la seqüència arribi a un extrem.
Posar la seqüència pel mig de la pantalla.

5.6 Trobar mitjançant un programa en C el màxim comú divisor de dos números naturals A y B emprant l'algorisme d'Euclides (empreu una funció amb pas de paràmetres A i B per valor).

```
Mentre A<>B,
    Si A>B Aleshores A← A - B
    Sino Si B > A Aleshores B← B - A
    FiSi
FiMentre
```

5.7 Implementar, l'algorisme anterior substituint la funció per la funció recursiva següent:

```
Mentre B<>0 Fer
(A,B) ← (B, mcd(A,B))
FiMentre
mcd = A
```

5.8 Donat un nombre fraccionari (A/B), simplificar la fracció.

Nota: L'algorisme esdevé trivial fent servir l'algorisme del mcd.

5.9 Fer un programa que endevini un numero pensat. Intentar realitzar un algorisme ràpid (per exemple, cerca dicotòmica, que endevina el número en un nombre $\log_2$(RANG) màxim de passos. Fer el programa modular, amb una rutina d'espera fins que el jugador hagi pensat el numero i una rutina d'error per detectar si el jugador l'ha enganyat.

5.10 Dibuixar un rectangle de mida (dx, dy) en posició (x0, y0) de la pantalla (limitar tots els números a valors inferiors a 20).

Ajudar-se d'una rutina que dibuixi una línia. En la rutina els paràmetres de la funció fan referència a: posició inicial, número de posicions, caràcters a dibuixar (esquerra, mig, dreta).

Nota: el codi ASCII proporciona els costats que fan que quedi ben dibuixat el rectangle. Dos possibles conjunts de caràcters són:

Línia doble:	Línia simple:
201,205,187	218,196,191
186, 32,186	179, 32,179
200,205,188	192,196,217

5.11 Realitzar un programa que mostri per pantalla les taules de multiplicar en base B. Per a la visualització correcta del resultat fer servir una rutina que realitzi el pas del número calculat en base 10 a base B.

Exercici per pensar.

5.12 Realitzar un algorisme que mostri com es mouen els discs en el problema de les Torres de Hanoi.

Les torres de Hanoi és un dels problemes clàssics de programació fàcilment resoluble emprant algorismes recursius. L'enunciat és el següent:

Es disposa de tres torres i n discs de mides diferents apilats de forma que els discs disminueixen de mida des del disc més petit al més gran en una torre, diem-ne (I)nicial, de forma que quan més gran és el disc més avall es troba en la torre. Es tracta de passar tots els discs (d'un en un) a la torre (F)inal de forma que mai cap disc de mida inferior té a sobre un disc de mida superior. Per això només es pot fer servir, addicionalment, la tercera torre anomenada (T)emporal.

Exercici 5.1

És fàcil veure que, en la sèrie de Fibonacci, cada terme és la suma dels dos termes anteriors. En conseqüència, tant el programa principal com la funció que calcula cada terme són simples de realitzar. El programa també mostra que les diferències del pas de variables per referència o per adreça són mínimes.

Pseudocodi: Programa principal

Entrades: n (nº de termes a generar)

Sortides: els n termes de la sèrie

```
Enter fib1, fib0, i        //Termes generadors, índex
Inici
      fib1 ← 1             //Inicialització
      fib0 ← 0
      Llegir (n)
      Si (n<2) Llavors Escriure ("Num no vàlid!")
      SiNo Llavors
         Escriure ("Fib1=",fib1)          //Primer terme
         Per (i=1;i<num) Fer
                  Escriure ("Fib"(i+1)"=",fibo(fib1,fib0))
                  i ← i+1
         FiPer
      FiSi
Fi
```

Entrades: Adreces termes generadors

Sortides: El terme i-èssim de la sèrie

Funció: Enter fibo (Enter &f1, &f0)

```
Enter temp      //terme temporal
Inici
         temp=f1;
         f1 += f0;
         f0=temp;
         return(f1);
Fi
```

Codificació en C

```
/* Càlcul de la sèrie de Fibonacci  */
#include <stdio.h>
int fibo(int&, int&);                    //Per adreça la declaració seria: int fibo(int*, int*);
void main()
{
         int num;
         int fib1=1, fib0=0;
         printf("Entra el nombre de termes: ");
         scanf("%i", &num);
         if (num<2) printf(" ...numero no valid!!\n");
         else
         {
                  printf("\tfib( 1) =    %i\n", fib1);
                  for (int i=1; i<num; i++)
                           printf("\tfib(%2i) = %4i\n", i+1, fibo(fib1, fib0));        //Pas per referència
                           // printf("\tfib(%2i) = %4i\n", i+1, fibo(&fib1, &fib0)); //Cas pas per adreça
         }
}
 //FUNCIÖ
 //Amb pas de paràmetres per referència:
 int fibo(int &f1, int &f0)
 {
          int temp;
          temp=f1;
          f1 += f0;
          f0=temp;
          return(f1);
 }
```

```
//Amb pas de paràmetres per adreça:
int fibo(int *f1, int *f0)
{
         int temp;
         temp=*f1;
         *f1 += *f0;
         *f0=temp;
         return(*f1);
}
```

Nota: Aquest exemple mostra clarament, la necessitat de passar les variables fib1 i fib0 per referència (o per adreça), ja que han de conservar el valor modificat en cada passada!

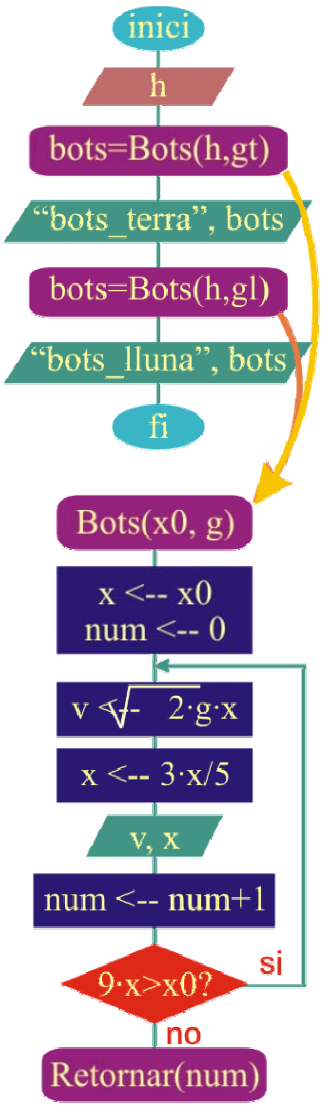
Exercici 5.2

El programa consta de la funció main() i de la funció de càlcul *bots*() que retorna el nombre de bots efectuats fins acomplir la condició. A a funció *bots*() se li passa l'alçada inicial i la gravetat.

Codificació en C

```
#include <stdio.h>
#include <math.h>
/*
Càlcul de la velocitat i alçada dels bots d'una pilota. Es compleix que v=sqrt(2gh). Considerar que
l'alçada següent ve donada per h'=3h/5. Calcular tots els bots d'alçada superior a 1/7 de l'inicial
*/
 const float gt=9.8;      //Constant de la gravetat sobre la terra
 const float gl=1.7;      //Constant de la gravetat sobre la lluna

 int Bots(float, float);  //Declaració de la funció de càlculs

 void main()
 {
        int bots;
        float h;

        printf("Entra h (m): ");                  //Entrada d'alçada
        scanf("%f", &h);
        printf("Bots en superficie terrestre: \n");
                                //Càlculs sobre superfície terrestre
        bots=Bots(h,gt);
        printf(" --> En total %u bots.\n\n", bots);
        printf("Bots en superficie llunar: \n");
                                //Càlculs sobre superfície llunar
        bots=Bots(h,gl);
        printf(" --> En total %u bots.\n\n", bots);
 }

 //Funció càlcul velocitat i alçada. Retorna el nombre de bots
 int Bots(float x0, float g)
 {
        int num=0;
        float v, x;

        x=x0;
        printf("-------v(m/s)-----h(m)--\n");      //Capçalera
        do
        {
                v=sqrt(2*g*x);                    //Càlcul velocitat
                x=3*x/5;                  //Càlcul alçada nou bot
                printf("\t%4.2f\tf%4.2f\n",v,x);
                num++;
        }while((9*x)>x0);                         //Condició de càlcul
        return(num);
 }
```

Diagrama de flux



Exercici 5.3

El sistema d'equacions s'estudia a partir del rang de la matriu (m) i la matriu adjunta(ma):

$$m = \begin{vmatrix} a & b \\ c & d \end{vmatrix},\ ma1 = \begin{vmatrix} e & b \\ f & d \end{vmatrix},\ ma2 = \begin{vmatrix} a & e \\ c & f \end{vmatrix}.$$

aleshores, si rang(m)=2: rectes secants → x=ma1/m, y=ma2/m

si rang(m)=1, rectes paral·leles si màxim_rang(m1 o m2) = 2, si no rectes coincidents.

si rang(m)=0, no té sentit el sistema d'equacions.

Diagrama de flux:

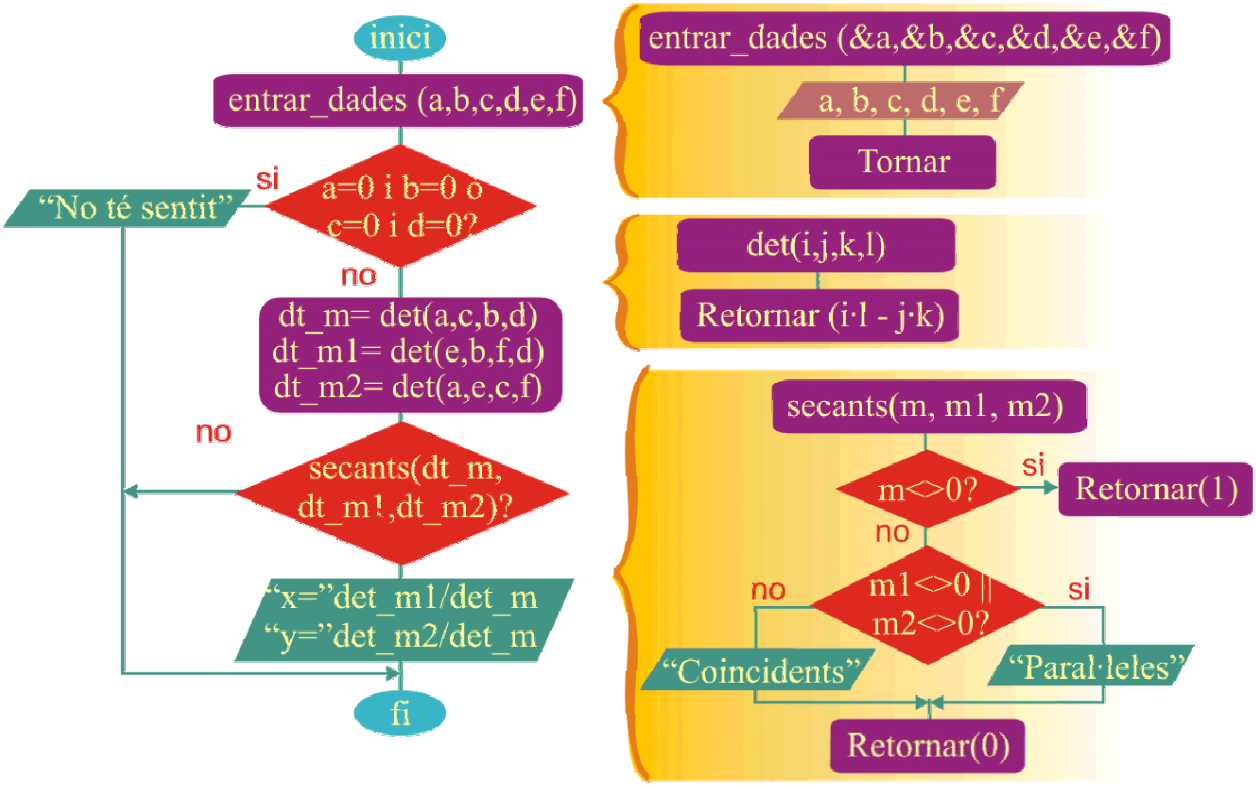
Es consideren 3 procediments/funcions:

void entrar_dades (float&, float&, float&, float&, float&, float&) → Entrada dels valors a, b, c, d, e, f del sistema d'equacions. Pas de variables per referència per a poder ser modificades.

float det(float, float, float, float) → Retorna el determinant d'ordre 2 de les variables passades. Pas de variables per valor.

int secants (float, float, float) → Analitza el sistema d'equacions i retorna un valor positiu quan les dues rectes són secants. Pas de variables per valor.

$$\left.\begin{matrix} a\cdot x + b\cdot y = e \\ c\cdot x + d\cdot y = f \end{matrix}\right\} \quad \begin{cases} m = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\ m1 = \begin{pmatrix} e & b \\ f & d \end{pmatrix} \\ m2 = \begin{pmatrix} a & e \\ c & f \end{pmatrix} \end{cases}$$



Codificació en C

```
/*
Resoldre el sistema d'equacions:
        ax + by =e
        cx + dy =f
*/
#include <stdio.h>

void entrar_dades(float&, float&, float&, float&, float&, float&);      //Entrada de dades
float det(float, float, float, float);                                  //Càlcul determinants
int secants (float, float, float);                                      //Estudi sistema d'equacions

void main()
{
        float a, b, c, d, e, f, det_m, det_m1, det_m2;
        entrar_dades(a, b, c, d, e, f);                         //Entrar dades: Pas per referència
        if ((a==0 && b==0) || (c==0 && d==0))                    //Comprovació sentit en dades
                printf("\nAixo es una pallassada!\n");
        else
        {
                det_m=det(a, c, b, d);                          //Càlcul determinants
                det_m1=det(e, b, f, d);
                det_m2=det(a, e, c, f);
                if (secants (det_m, det_m1, det_m2))            //Estudi rangs
                {
                        printf("\tx = %3.2f\n", det_m1/det_m);    //Solució rectes secants
                        printf("\ty = %3.2f\n", det_m2/det_m);
                }
        }
}

void entrar_dades(float &a, float &b, float &c, float &d, float &e, float &f)   //Entrada de dades
{
        printf("RESOLUCIÓ D'UN SISTEMA D'EQUACIONS\n");
        printf("\ta x + b y = e\n");
        printf("\tc x + d y = f\n");
        printf("Entra a, b, e: ");
        scanf("%f%f%f", &a, &b, &e);
        printf("Entra c, d, f: ");
        scanf("%f%f%f", &c, &d, &f);
}
float det(float i, float j, float k, float l)                           //Càlcul determinants
{
        return(i*l-j*k);
}
int secants(float m, float m1, float m2)                                //Estudi sistema d'eqüacions
{
        if (m!=0) return(1);                                            //Rectes secants
        else
        {
                if (m1!=0 || m2!=0) printf("Rectes paral.leles\n");     //Rectes paral.leles
                else printf("Rectes coincidents\n");                   //Rectes coincidents
                return(0);
        }
}
```

Exercici 5.4

Programa fet el màxim de simple per a entendre el funcionament. El programa constarà de la funció main() i dos rutines: *imprimir*( ) i *bucle_espera*( ).

Funció main(). És un bucle infinit que calcula el valor *a* (potència de 2) a imprimir cada passada, i crida rutina d'espera. a, inicialitzat a A=0x80 (= 2^8), es divideix per 2 durant 8 passades, fent córrer el 1 cap a la dreta.

La rutina *imprimir*( ) imprimeix, en binari, el valor passat (potència de 2) emprant la instrucció case.

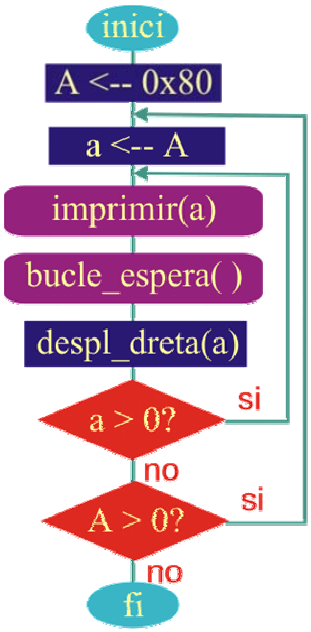
El bucle d'espera s'ha fet emprant l'operació increment durant un cert nombre de bucles. En el proper exercici s'utilitzarà funcions específiques temporals de la llibreria time.h

Pseudocodi:

Funció main():

Entrades: cap. S'utilitza un valor inicial A=0x80 = "10000000"
Sortides: S'imprimeix la variable a, obtinguda per desplaçament a dreta d'A (=divisió per 2)

```
Enter A, a                  // Variable inicial, variable comptadora
Inici
A ← 0x80
Fer
        a ← A
        Fer
                imprimir(a)     //Imprimir a (un 1 entre 0's)
                bucle_espera()//Temps d'espera
                a ← a>>1        //Desplaçament a dreta
        Mentre (a>0)            //Bucle intern que fa córrer el 1
Mentre (A>0)                    //Bucle infinit
Fi
```



Procediment imprimir( ):

Entrades: rep a (8 bits), un enter amb un 1 en només un lloc
Sortides: S'imprimeix el valor d'a en binari

```
Procediment imprimir (Enter a)
Inici
        EnCasDe (a)
                0x80: Escriure ("\r10000000"). SortirCas
                0x40: Escriure ("\r01000000"). SortirCas
                0x20: Escriure ("\r00100000"). SortirCas
                0x10: Escriure ("\r00010000"). SortirCas
                0x08: Escriure ("\r00001000"). SortirCas
                0x04: Escriure ("\r00000100"). SortirCas
                0x02: Escriure ("\r00000010"). SortirCas
                0x01: Escriure ("\r00000001").
        FiEnCasDe
Fi
```

Procediment bucle_espera( ):

Entrades/sortides: No n'hi ha.

```
Procediment bucle_espera
Enter t
Inici
        t ← 0
        Fer
                t ← t+1
        Mentre (t<20000000)
Fi
```

Codificació en C

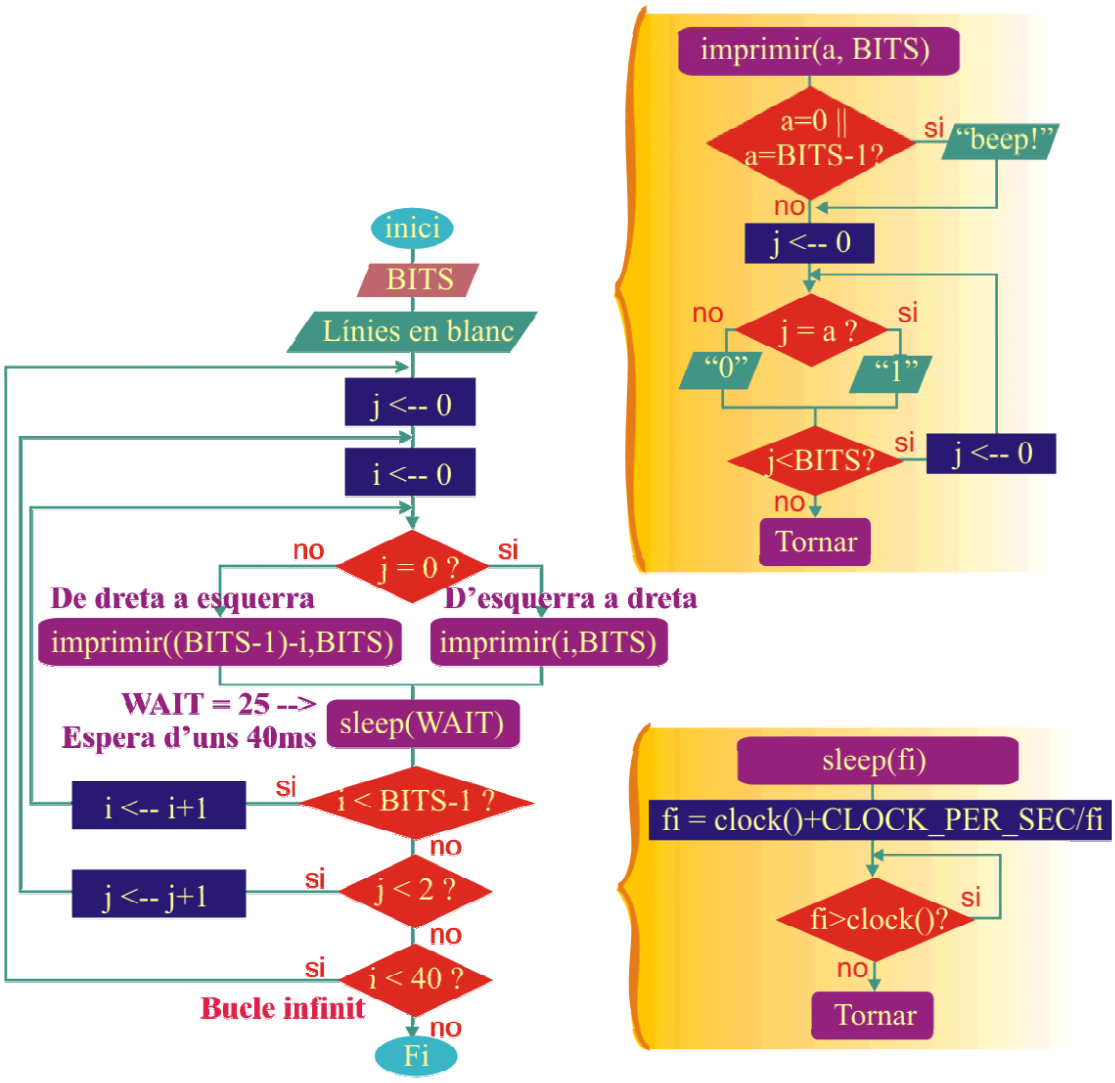
```
/*
Fer navegar un 1 entre 8 zeros, d'esquerra a dreta: 10000000-> 01000000->....->00000001-
>10000000->01000000->... Cas bàsic emprant un bucle 'casolà' d'espera
*/
#include <stdio.h>
void bucle_espera(void);                //Funció d'espera
void imprimir(int);                     //Funció d'impressió

void main()
{
        int A=0x80, a;
        do
        {
                a=A;
                do
                {
                        imprimir(a);            //Imprimeix sobre la mateixa línia
                        bucle_espera();         //bucle espera per visualització
                        a = a>>1;               //desplaçament
                }while (a>0);
        }while(A>0);
        printf("\n");
}

void imprimir(int a)
{
        switch(a)
        {
        case 0x80: printf("\r10000000"); break;    //printf("\r"); → Retorna al començament de línia
        case 0x40: printf("\r01000000"); break;
        case 0x20: printf("\r00100000"); break;
        case 0x10: printf("\r00010000"); break;
        case 0x08: printf("\r00001000"); break;
        case 0x04: printf("\r00000100"); break;
        case 0x02: printf("\r00000010"); break;
        case 0x01: printf("\r00000001");
        }
}

void bucle_espera(void)
{
        int t=0;
        do {}while(t++<20000000);
}
```

Exercici 5.5

És una continuació de l'exercici anterior, posant èmfasis en els detalls. Entre aquests es consideren:

Permetre entrar el nombre de bits a visualitzar.
La posició del 1 es calcularà a partir d'una variable (no cal emprar case).
Fer el bucle d'espera emprant funcions temporals que proporciona la llibreria *time.h*.
Generar un *beep* cada cop que la seqüència arribi a un extrem.
Posar la seqüència pel mig de la pantalla.

Diagrama de flux.



Codificació en C.

```
/*
Generador de seqüència per n bits. Exem cas n=4: 1000-> 0100->0010->0001->0010->0100->...
És interessant entendre el funcionament del bucle d'espera basat en la funció clock() que retorna el nombre de períodes passats del processador. La duració d'un puls és, aproximadament, 1/CLOCKS_PER_SEC.
*/

#include <stdio.h>
#include <time.h>
#define WAIT 25                                  //Temps de desplaçament 1/WAIT segons

void sleep(clock_t);
void imprimir(int, int);

void main()
{
        int i, j, BITS;
        printf("Entra el nombre de bits: ");     //Entrar el nombre de bits
        scanf("%i", &BITS);

        for (i=0; i<7; i++) printf("\n");        //Deixar línies en blanc

        do                                       //PROGRAMA PRINCIPAL
        {
                for (j=0; j<2; j++)              //j=1->anar, tornar<-j=0
                        for (i=0; i<(BITS-1); i++)   //7 transicions
                        {
                                if (!j) imprimir(i, BITS);   //i indica posicio del 1
                                else imprimir((BITS-1)-i, BITS);
                                sleep(WAIT);             //bucle espera en seg/WAIT
                        }
        }while(i<40);                            //COMPTA: bucle infinit!
        printf("\n");
}

void imprimir(int a, int BITS)                   //Rutina impressio bits
{
        int j;
        printf("\r\t\t");                        //Retorn a principi de linia
        if (a==0 || a==(BITS-1)) printf("\a");   //Beep!
        for (j=0; j<BITS; j++)
                if (j==a) printf("1");
                else printf("0");
}

void sleep(clock_t fi)                           //Temps d'espera (invers. prop. a fi)
{
        fi = clock()+CLOCKS_PER_SEC/fi;
        while(fi>clock());
}
```

Exercici 5.6

El resultat és immediat.

La solució consta del programa principal, que fa el tractament de les dades d'entrada i de la funció de càlcul del mcd.

Pseudocodi: Programa principal

Entrades: 2 enters

Sortides: el mcd

```
Enter a, b, mcd
Inici
      Llegir (a, b)
      Si (a=0 ò b=0) Llavors
         Escriure("El 0 no té divisor!")
      SiNo Llavors
         mcd=euclides(a, b)
         Escriure(mcd)
      FiSi
Fi
```

Entrades: Els dos enters a i b

Sortides: Un enter: el mcd

Funció: Enter euclides (Enter a, b)

```
Inici
      Mentre (a<>b);
         Si (a>b) Llavors a=a-b
         Sino Llavors b=b-a
         FiSi
      FiMentre
      Retornar(a)
Fi
```

Codificació en C

```
/*
Calcul del mcd
*/
#include <stdio.h>

int euclides(int a, int b);                                    //Funció càlcul mcd

void main()
{
        int a,b, mcd;
        printf("METODE D'EUCLIDES\n");
        printf("  Entra dos enters a, b: ");                   //Entrada dels dos enters
        scanf("%i%i", &a, &b);
        if (a==0 || b==0) printf("        --> El 0 no te divisors!!\n");   //Comprovació de dades
        else
        {
                mcd = euclides(a,b);                           //Crida de la funció
                printf(" --> mcd(%i, %i) = %i\n", a, b, mcd);  //Càlcul del mcd
        }
}

int euclides(int a, int b)                                     //Funció càlcul mcd
{
        while (a != b) (a>b)?(a -= b):(b -= a);
        return (a);                                            //Retorna el mcd
}
```

Exercici 5.7

És com l'exercici anterior. Només canvia l'algorisme emprat per la funció euclides( ), que és el següent:

Entrades: Els dos enters a i b

Sortides: Un enter: el mcd

Funció: Enter euclides_rec (Enter a, b)

```
Inici
    Si (b=0) Llavors Retornar(a)
    Sino Llavors Retornar(euclides_rec(b, a%b))
    FiSi
Fi
```

Codificació en C

```
/*
Calcul del mcd
*/
#include <stdio.h>

int euclides_rec(int a, int b);                                  //Funció càlcul mcd

void main()
{
        int a,b, mcd;
        printf("METODE D'EUCLIDES\n");
        printf("  Entra dos enters a, b: ");                     //Entrada dels dos enters
        scanf("%i%i", &a, &b);
        if (a==0 || b==0) printf("        --> El 0 no te divisors!!\n");   //Comprovació de dades
        else
        {
                mcd = euclides_rec(a,b);                         //Crida de la funció
                printf(" --> mcd(%i, %i) = %i\n", a, b, mcd);    //Càlcul del mcd
        }
}

int euclides_rec(int a, int b)                                   //Funció càlcul mcd
{
        if (b == 0) return (a);
        else return (euclides_rec(b,a%b));
}
```

Exercici 5.8

Per a simplificar una fracció només cal trobar el mcd del numerador i del denominador i dividir tant el numerador com el denominador pel mcd.

Per tant: la funció ja la coneixem. El programa principal té poques variacions respecte a l'emprat per trobar el mcd: ara només cal dir que hem d'entrar dos enters que corresponen al numerador i denominador d'una fracció...

Pseudocodi: Programa principal

Entrades: 2 enters, corresponents al numerador i al denominador de la fracció

Sortides: 2 enters, corresponents al numerador i al denominador de la fracció simplificada

```
Enter a, b, mcd, as, bs          //fracció no simplificada (a/b), mcd, fracció simplificada (as/bs)
Inici
      Llegir (a, b)
      Si (b=0) Llavors Escriure("El 0 no té divisor!")
      SiNo Si (a=0) Llavors Escriure("Això és 0!")
      SiNo Llavors
         mcd=euclides(a, b)
         as=a/mcd
         bs=b/mcd
         Escriure("a/b=",as,"/",bs)
      FiSi
Fi
```

Codificació en C

```
/*
Simplificació de fraccions
*/
#include <stdio.h>
int euclides_rec(int a, int b);
void main()
{
        int a,b, mcd, as, bs;
        printf("SIMPLIFICACIO DE FRACCIONS\n");
        printf("  Entra el numerador i el denominador (A/B): ");
        scanf("%i%i", &a, &b);
        if (b==0) printf("\t ...el 0 no te divisors!!\n");
        else if (a==0) printf("\t ...aixo es 0!!\n");
        else
        {
                mcd = euclides_rec(a,b);
                as=a/mcd;
                bs=b/mcd;
                printf("\tA/B = %i/%i = %i/%i\n ", a, b, as, bs);
        }
}
int euclides_rec(int a, int b)
{
        if (b == 0) return (a);
        else return (euclides_rec(b,a%b));
}
```

Exercici 5.9

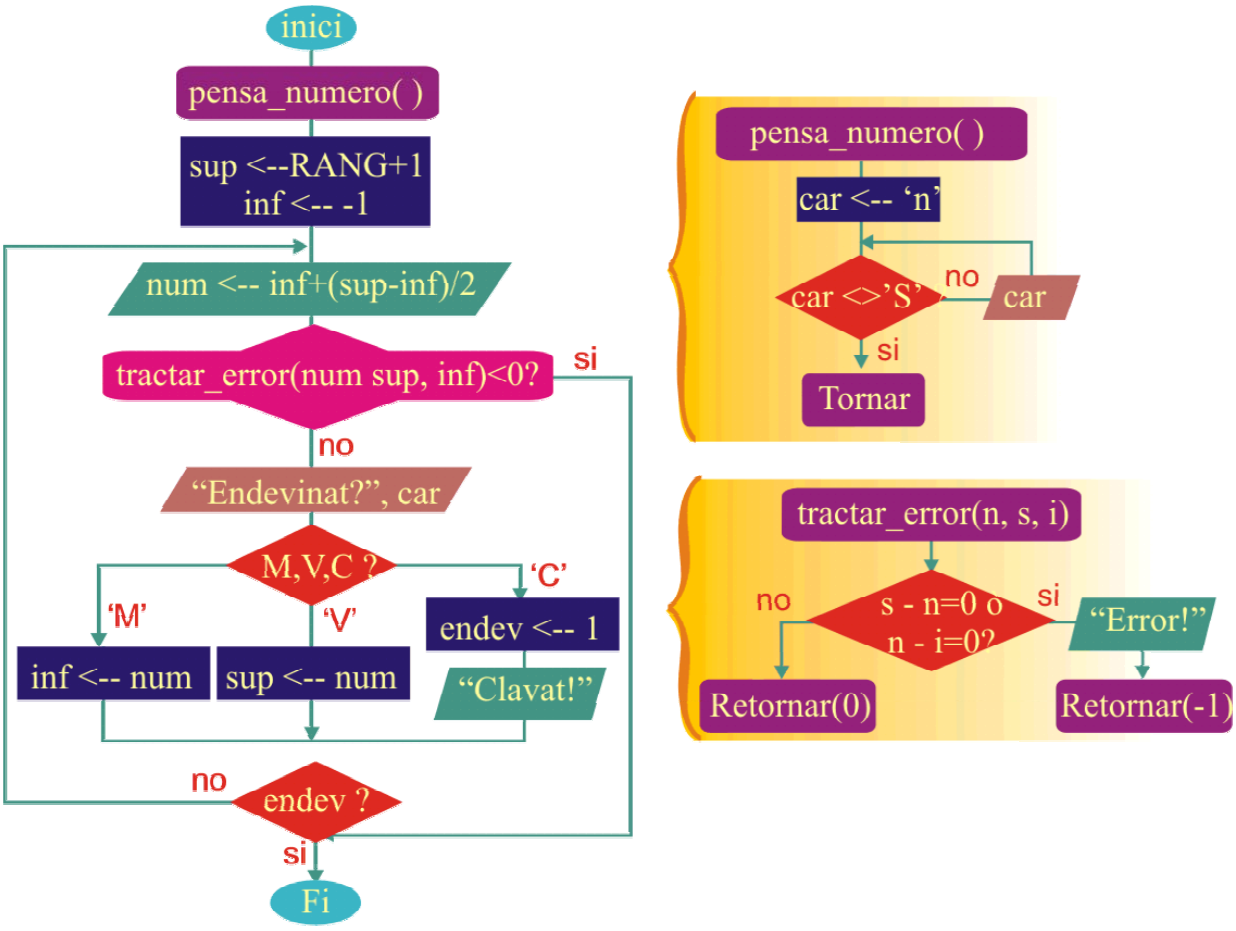
L'algorisme de cerca dicotòmica va dividint l'espai de números per la meitat, d'acord amb la resposta que es doni. L'algorisme sempre dirà el número mig reduirà l'espai de números d'acord amb un límit sup(erior) i un altre inf(erior). Si la resposta és més avall, farà que el límit superior sigui el nombre jugat; si la resposta és un número superior, aleshores posarà el nombre jugat com a límit inferior.

Per exemple, si suposem un RANG=25 i pensem el nombre 18, aquests límits serien:

resposta	inferior	superior	nova tirada
	0	25	12
amunt	12	25	19
avall	12	19	15
amunt	15	19	17
amunt	17	19	18
clavat!			

Diagrama de flux.

El programa constarà del programa principal i dues funcions. La funció *pensa_numero( )* és un procediment d'espera a què l'usuari digui que ha pensat el número. La funció *tractar_error( )* retorna un número negatiu quan l'usuari ha enganyat al programa en les seves contestes.



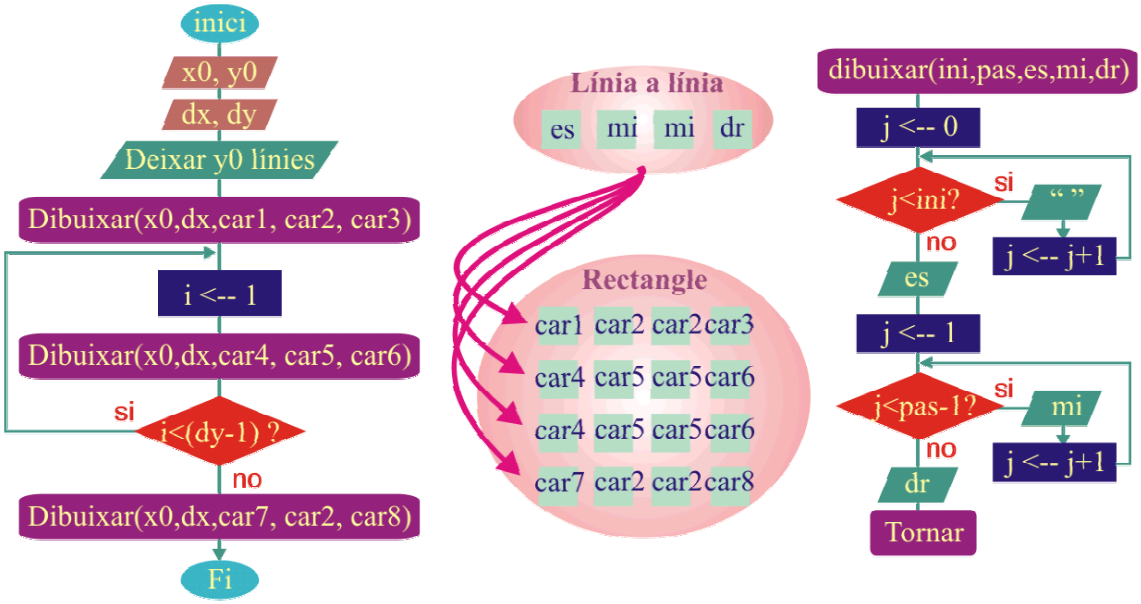
Codificació en C

```
/* Endevinar el numero pensat en un nombre màxim de passos log2(RANG).*/
#include <stdio.h>
#define RANG 1000                       // Rang de números
void pensa_numero(void);                //Rutina d'espera fins a haver pensat el número
int tractar_error(int, int, int);       //Comprovació de si s'ha enganyat al programa
int main()
{
        char car='n';
        int num, sup, inf, endev=0;
        pensa_numero();
        sup=RANG+1;
        inf=-1;
        do
        {
                num=inf+(sup-inf)/2;
                if (tractar_error(num, sup, inf)) return (-1);
                printf("\t... es el %i (a(m)unt, a(v)all, (c)lavat)?", num);
                fflush(stdin);
                scanf("%c", &car);
                car = car & 0xdf;               //Conversió a majúscules
                switch (car)
                {
                        case 'M':inf=num; break;
                        case 'V':sup=num; break;
                        case 'C': printf("\n\n\t\t...ENDEVINAT...\n\n");
                                endev=1;
                };
                printf("\n");
        }while (!endev);
        return(0);
}
void pensa_numero(void)                 //Rutina d'espera fins a haver pensat el número
{
        char car = 'n';
        while (car!='S')
        {
                printf("Pensa un numero entre 0 i %i. Ja ho has fet (s/n)?", RANG);
                fflush(stdin);
                scanf("%c",&car);
                car = car & 0xdf;               //Conversió a majúscules
        }
}
int tractar_error(int n, int s, int i)          //Comprovació de si s'ha enganyat al programa

{
        if ((s-n)==0 || (n-i)==0)
        {
                printf("\n\tA QUI ENGANYES...?\n\n");
                return(1);
        }
        else return(0);
}
```

Exercici 5.10

És un exemple simple que mostra com es poden fer servir els caràcters ASCII per a donar una imatge més elegant als programes. Fa ús de caràcters, es podria dir 'geomètrics' de la taula ASCII per dibuixar un rectangle de tamany dx * dy a partir de la posició (x0, y0) de la pantalla. El programa s'utilitza una rutina de dibuix de cada línia que rep, com a paràmetres, la posició i mida a dibuixar i els caràcters dels costats i del mig a emprar.

Diagrama de flux.



Codificació.

```
/* Dibuixar d'un rectangle de tamany (dx, dy) en la posició (x0, y0) de la pantalla. */
#include <stdio.h>

void dibuixar(int, int, int, int, int);
void main()
{
	int x0, y0, dx, dy, i;
	do                                                    //Entrar posició inicial del rectangle
	{
		printf("Dona posicio inicial (x0,y0): ");
		scanf("%i%i", &x0, &y0);
	}while(x0<0 || y0<0);
	do                                                    //Entrar tamany del rectangle
	{
		printf("Dona tamany rectangle (dx,dy): ");
		scanf("%i%i", &dx, &dy);
	}while(dx<2 || dy<2);
	for (i=0; i<y0; i++) printf("\n");                    //Dibuix del rectangle
	dibuixar(x0, dx, 218, 196, 191);
	for (i=1; i<(dy-1); i++) dibuixar(x0, dx, 179, 32, 179);
	dibuixar(x0, dx, 192, 196, 217);
}
void dibuixar(int ini, int pas, int es, int mi, int dr)   //Dibuix d'una línia
{
	int j;
	for (j=0; j<ini; j++) printf(" ");
	printf("%c", es);
	for (j=1; j<(pas-1); j++) printf("%c", mi);
	printf("%c\n",dr);
}
```

Exercici 5.11

El programa consta de la funció main() i de la funció bB(), que converteix un número a base B. El programa principal no té cap secret: és un doble bucle que calcula les taules de multiplicar i posa la sortida en format elegant!. La tasca de conversió la fa la funció bB( ), que no fa res més que realitzar el canvi de base:

> **d**ivident = **quo**cient * **B**ase + **res**ta, on **res** són els dígits base B que es van trobant.

El programa està ben documentat per entendre els passos principals de l'algorisme.

Codificació en C

```
/*
Taules de multiplicar base B amb rutina de visualització.
*/
#include <stdio.h>

int bB(int, int);                                       //Rutina de visualització en base DIM
void main()
{
        int i, j, B;                                                                    //índexs i Base
        printf("Entra la base: ");
        scanf("%i", &B);
        printf("\nTaules multiplicacio del %i:\n", B);
        for (i=0;i<B+1;i++)                                                             //Bucle extern
        {
                for (j=0;j<B+1;j++)     printf("\t%i * %i = %i\n", i, j, bB(i*j, B));       //Bucle intern
                printf("\n");
        }
}
int bB(int d, int B)       //Conversió a base B per a visualització correcta. Paràmetres: dividend, base
{
        int quo, res=0, i=1;     //quocient, resta i índex de posició: el 0 és la posició més a la dreta
        do
        {
                quo=d/B;                        //Es troba el quocient
                res += (d-quo*B)*i;             //resta = (dividend-quocient*Base)*posició
                d=quo;                          //el quocient serà el nou dividend
                i *= 10;          //Cada índex ocupa una posició més a la dreta en la visualització
        }while (d>0);                           //La conversió s'acaba quan el dividend es fa 0.
        return(res);
}
```

Exercici 5.12

Resoldre el problema de les torres de Hanoi és un dels típics exemples que recursivament es pot 'arribar a entendre amb certa facilitat'.

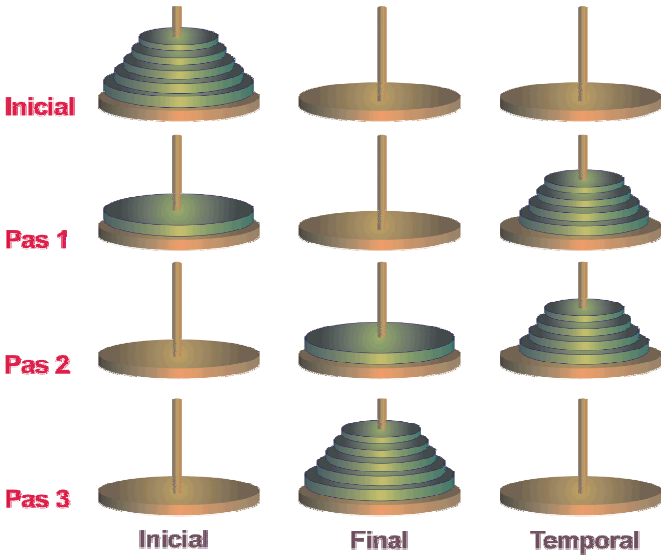
La seva solució passa per entendre tres passos fonamentals, que són els que posteriorment es sintetitzen en l'algorisme:

i) Primer es transporten (n-1) discs de la torre inicial a la temporal

ii) Després es transporta el disc que queda en la torre inicial a la final

iii) Finalment es transporten els (n-1) discs de la torre temporal a la final.

Realitzant l'algorisme de forma recursiva s'aconsegueix transportar tots els discs de la torre inicial a la torre final de forma que mai un disc de tamany superior està per sobre d'un de tamany inferior.



L'algorisme mostra com es mouen els discs.

Codificació en C

```
/*
Torres de Hanoi
 */

#include <stdio.h>

void hanoi (int n, char i, char f, char t);

void main(void)
{
	int n;
	printf("\t--- TORRES DE HANOI ---\n");
	printf("Entra el nombre de discs: ");
	scanf("%i", &n);
	hanoi(n, 'I', 'F', 'T');
}

void hanoi (int n, char i, char f, char t)
{
	if (n==1) printf("Disc 1 de torre %c a torre %c\n", i, f);
	else
	{
		hanoi(n-1, i, t, f);
		printf("Disc %i de torre %c a torre %c\n", n, i, f);
		hanoi(n-1, t, f, i);
	}
}
```