Problemes capítol 6

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.

6.1 Implementar en C una funció que examini un array unidimensional i retorni **1** si està ordenat i **0** si no ho està.

6.2 Una estació climàtica proporciona dues temperatures diàries (màxima i mínima) durant una sèrie de dies. Es demana implementar l'algorisme que imprimeixi:

- Les temperatures mitges màxima i mínima.
- El dia de màxima, amb la corresponent màxima.
- El dia de mínima, amb la corresponent mínima.

Nota: Considereu que les temperatures màxima i mínima d'una sèrie de dies estan emmagatzemades en els dos vectors max[n] i min[n], n = número de dies.

6.3 Es té una bicicleta amb un canvi Shimano que consta de tres plats i 7 pinyons. Els plats tenen 24, 32 i 42 dents. Els pinyons tenen 28, 24, 21, 19, 17, 15 i 13 dents. Dóna totes les relacions de canvi possibles.

Si la roda té un radi de 34cm, dóna l'avanç (en m.) que es produeix a cada pedalada.

6.4 Realitzar un programa que torni el canvi donada una quantitat a retornar. La moneda de què es disposa són bitllets de 500€, 200€, 100€, 50€, 20€, 10€, i 5€. Considerar que el canvi a retornar és múltiple de 5€.

6.5 Emprant l'algorisme del mcd(a, b), trobar el mcd d'un array de n nombres.

6.6 Detecció dels primers N nombres primers que s'emmagatzemaran en un array de N posicions.

Considerar un algorisme que fa córrer un índex i de l'array on s'emmagatzemen fins que es detecten els N primers. Per a cada i es comprova (amb cadascun dels nombres primers emmagatzemats) si és divisible per un d'ells. De no ser-ho, el nombre és un nou nombre primer i s'ha de posar en l'array de primers.

6.7 Sobre diagonals en matrius. Fer un programa que creï una matriu quadrada de dimensió n x n amb els valors de totes les diagonals iguals

a) Fer que les diagonals iguals són les que vagin de dreta a esquerra.

b) Ara aplicar-ho a les diagonals esquerra-dreta.

c) Donada la posició d'un àlfil en un tauler d'escacs, doneu totes les caselles a on pot moure.

6.8 Sobre permutacions.

a) Fer un programa que escrigui per pantalla totes les permutacions amb repetició possibles amb 3 lletres no repetides.

b) Fer ara l'algorisme que dongui totes les permutacions sense repetició possibles.

6.9 Entrar una frase i comptar els cops que cada vocal apareix repetida en la frase.

6.10 Dissenyar en C un algoritme que prengui dues cadenes i determini si el contingut de la primera es troba a la segona.

6.11 Càlcul del determinant d'una matriu de 3x3

$$\begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix} = a_{00}.\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} + a_{01}.\begin{vmatrix} a_{12} & a_{10} \\ a_{22} & a_{20} \end{vmatrix} + a_{02}.\begin{vmatrix} a_{10} & a_{11} \\ a_{20} & a_{21} \end{vmatrix},$$

coneixent que: $\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}.a_{22} - a_{12} * a_{21}$

6.12 Disposem de la següent estructura de dades (segons declaració en C) **char dades[40];** on volem implementar una llista circular de caràcters (es a dir treballem amb el vector com si després de **dades[39]** s'hi trobés **dades[0]**. El vector està carregat amb caràcters els valors dels quals desconeixem.

Dissenyeu una funció que treballi amb el vector i que quan trobi un '*' canviï els caràcters anterior i posterior (llevat que siguin '*') per un '?', i substitueixi l'asterisc per un '@'.

Exemple, suposant que el vector tingués 20 posicions):

Abans

A	&	*	/	1	*	*	*	A	D	H	J	J	*	G	J	P	*	.	*

Després

?	?	@	?	?	@	@	@	?	D	H	J	?	@	?	J	?	@	?	@

6.13 GENERADOR DE NOMBRES ALEATORIS. Realitzar un histograma de nombres aleatoris entre 0 i 50.

Explicació: La generació de nombres aleatoris es fa emprant la funció *rand()* de la llibreria *stdlib.h* que dóna una sortida *float*. La sortida està limitada per la constant *RAND_MAX* que en limita el nombre major. Per tant, dividint la sortida per aquesta constant es normalitza entre 0 i 1.

Per aconseguir una sortida realment aleatòria cada cop que s'executa el programa s'ha d'emprar una llavor inicial que s'obté emprant la funció *srand(time(NULL))*. Aquesta variable temporal permet variar la condició inicial generadora.

Exercicis més llargs (però molt interessants).

6.14 Crear un array de N enters des del conjunt buit (n=0) amb el següent conjunt de funcions que es podran realitzar amb l'array:

- Cercar enter
- Eliminar enter
- Insertar enter (sempre i quan no existeixi)
- Veure l'array
- Sortir

El programa principal ha de permetre triar l'operació a realitzar i l'array tindrà els noms ordenats de menor a major.

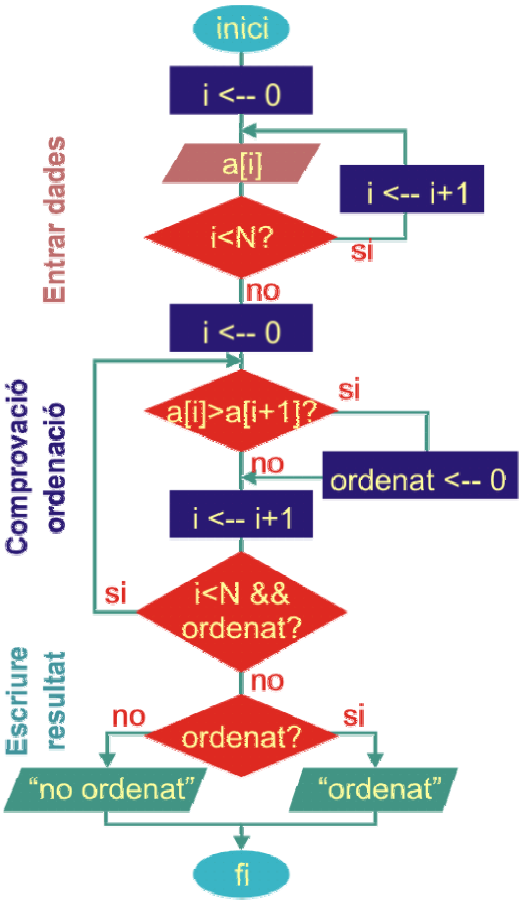
6.15 Col·locar n reines en un tauler de n x n caselles (n>3).

Exercici 6.1

Diagrama de flux:

Entrades: un array de 10 valors entrat per teclat

Sortides: Si l'array està ordenat o no



Codificació en C

```
/*
Veure si un array està ordenat
*/
#include <stdio.h>
#define N 10
void main()
{
	int i, a[N], ordenat=1;
	printf("Entra 10 numeros: ");
	for (i=0; i<N; i++) scanf("%i", &a[i]);		//Entrar les dades de l'array
	i=0;
	do
		if (a[i]>a[i+1]) ordenat=0;		//Comparació de valors
	while (++i<(N-1) && ordenat);
	if (ordenat) printf("--> Array ordenat\n");	//Imprimir resultat
	else printf("--> Array NO ordenat\n");
}
```

Exercici 6.2

Es creen dos arrays: un per les temperatures màximes i un altre per les mínimes.

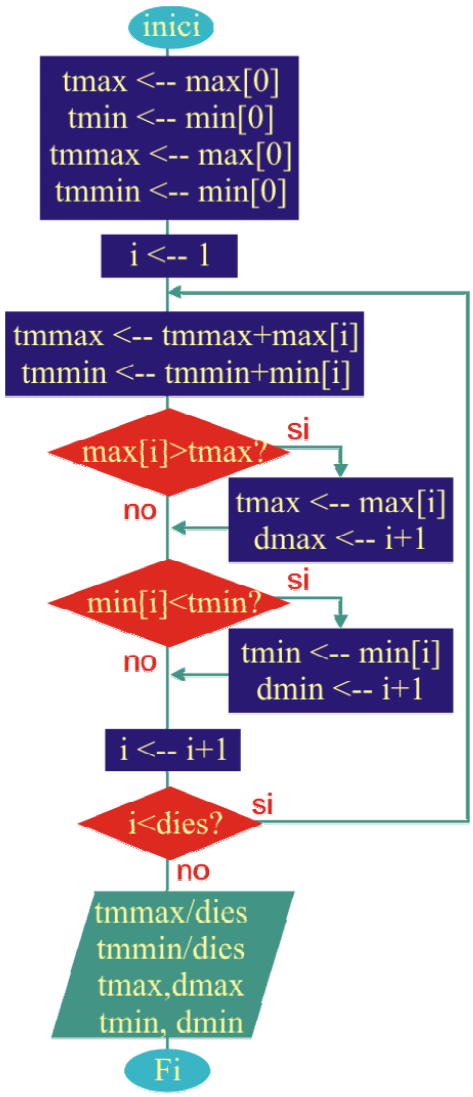
Les temperatures en els arrays, es poden posar en inicialitzar l'array (més simple, com es fa aquí) o es poden demanar per pantalla.

Recorrent amb un índex i els arrays es troben les temperatures màxima, mínima i les mitges.

Codificació en C

```
/* Càlcul temperatures d'una estació metereològica */
#include <stdio.h>
#define dies 10

float max[dies] = {10, 11, 13, 12, 16, 4, 15, 18, 15, 10};
float min[dies] = {0, 1, 3, 2, 6, -4, 5, 8, 5, 0};

void main()
{
	int dmax=1, dmin=1;
	float tmmax, tmmin, tmax, tmin;

	tmax = max[0];			//Inicialització
	tmin = min[0];
	tmmax = max[0];
	tmmin = min[0];

	for (int i=1; i<dies; i++)	//Càlculs
	{
		tmmax += max[i];
		tmmin += min[i];
		if (max[i]>tmax)
		{
			tmax = max[i];
			dmax = i+1;
		}
		if (min[i]<tmin)
		{
			tmin = min[i];
			dmin = i+1;
		}

	}
					//Imprimir resultats
	printf("\tTmitja maxima = %3.1fC\n", tmmax/dies);
	printf("\tTmitja minima = %3.1fC\n", tmmin/dies);
	printf("\tTmaxima %3.1fC el dia %i\n", tmax, dmax);
	printf("\tTminima %3.1fC el dia %i\n", tmin, dmin);
}
```

Diagrama de flux:



Exercici 6.3

Pseudocodi:

Entrades: 2 arrays: array relació de plats i array relació de pinyons

Sortides: Impressió de tots els canvis possibles i les distàncies recorregudes

```
Enter pl[3] = {24, 32, 42}, pi[7]={28, 24, 21, 19, 17, 15, 13}//Relació de plats, relació de pinyons
Const Real PI=3.141592, R=0.34              //Constants PI, radi de la roda (m)
Real canvi, l                               //Variable relació de canvi, distància recorreguda
Inici
      Per (i=0;i<3) Fer
         Per (i=0;i<7) Fer
                  canvi=pl[i]/pi[i]
                  l=2·PI·R·canvi
                  Escriure(pl[i]:pi[i]"→"canvi,l)
                  j ← j+1
         FiPer
         i ← i+1
      FiPer
Fi
```

Codificació en C

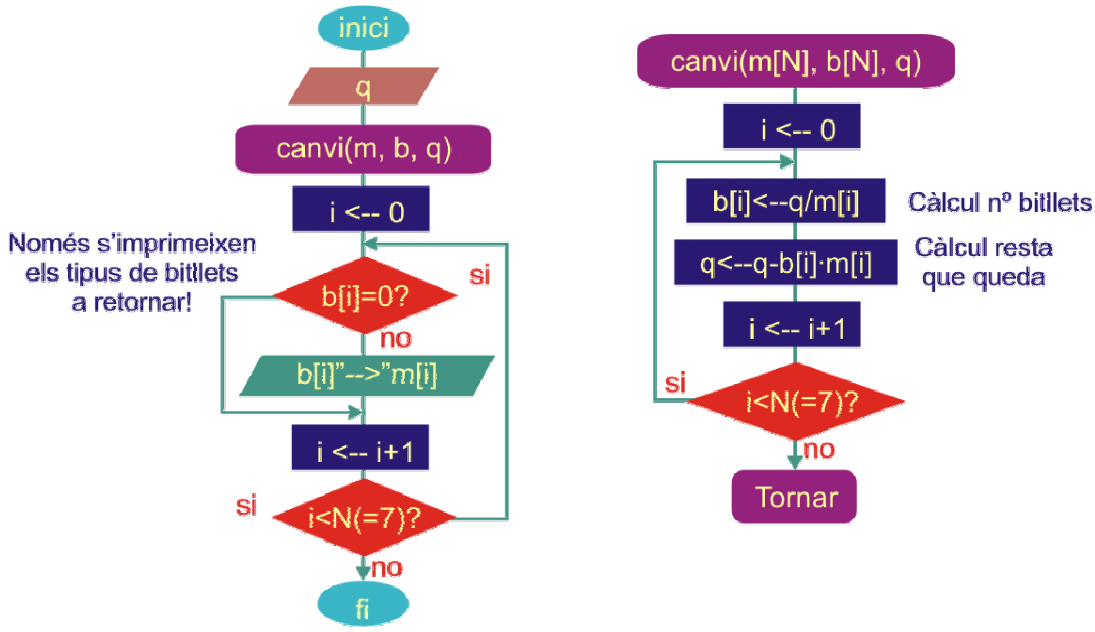
```
/*
Donar la relacio de canvi d'una bicicleta. La bicicleta té una relació de canvi que consta de
        3 plats: 24, 32 i 42 dents
        7 pinyons: 13, 15, 17, 19, 21, 24 i 28 dents
*/
#include <stdio.h>
#define PI 3.141592
#define R 0.34
void main()
{
        int pl[]={24, 32, 42};                          //Relació de plats
        int pi[]={28, 24, 21, 19, 17, 15, 13};          //Relació de pinyons

        float canvi, l;
        for (int i=0; i<3; i++)                         //Per a tots els plats
        {
                for(int j=0; j<7; j++)                  //Per a tots els pinyons
                {
                        canvi=float(pl[i]*1.0/pi[j]);   //Trobar la relació de canvi
                        l=2*PI*R*canvi;
                        printf("%i:%i=%2.1f-%2.1fm ", i, j, canvi, l);          }
                printf("\n");
        }
}
```

Exercici 6.4

La funció main() demana que s'introdueixi la quantitat a retornar, crida a la funció de càlcul de canvi i imprimeix el resultat.

La funció canvi rep com a paràmetres els arrays b[ ] → moneda existent (valor dels bitllets de canvi), m[ ] → moneda a retornar (on s'hi posa la quantitat de bitllets a retornar de cada valor) i la quantitat (q) a retornar.

Diagrama de flux:



Codificació en C

```
/*
Canvi de moneda
*/
#include <stdio.h>
#define N 7
void canvi(int [], int [], int);
void main()
{
	int i, q;                                       //Quantitat a tornar canvi
	int m[]={500, 200, 100, 50, 20, 10, 5};         //Moneda existent
	int b[]={0, 0, 0, 0, 0, 0, 0};                  //Bitllets a tornar de cada tipus
	printf("Entra quantitat: ");
	scanf("%i", &q);
	canvi(m, b, q);
	for (i=0; i<N; i++)
		if (b[i]!=0) printf("\t%i bitllets de %i\n", b[i], m[i]);      //Impressió resultat
	printf("\n");
}
void canvi(int m[N], int b[N], int q)                   //Funció canvi
{
	for (int i=0;i<N;i++)
	{
		b[i]=q/m[i];
		q -=b[i]*m[i];
	}
}
```

Exercici 6.5

La funció main()controla el flux de números amb els que es realitza cada càlcul del mcd(a, b).

La funció mcd(a, b) retorna el mcd de cada dos nombres.

L'algorisme no té cap secret. La funció mcd(a, b) és la mateixa ja implementada en el capítol 5.

Pseudocodi:

Entrades: n enters entrats per teclat

Sortides: El mcd(dels n enters)

```
Const Enter N=5                    //Número d'enters a trobar el mcd
Enter a[N], mcd, i                 //Array d'enters, mcd, índex
Inici
    Per (i=0; i<n) Fer             //Lectura dels nombres
        Llegir(a[i])
        i ← i+1
    FiPer
    mcd ← a[0]                     //Càlcul del mcd
    Per (i=1; i<n) Fer
        mcd ← euclides(mcd, a[i])
        i ← i+1
    FiPer
    Escriure(mcd)                  //Escriure el mcd
Fi
```

Codificació:

```
/*
Calcul del mcd
*/
#include <stdio.h>
#define N 5
int euclides(int, int);                         //Funció càlcul mcd
void main()
{
    int a[N], mcd, i;
    printf("METODE D'EUCLIDES\n");
    for (i=0; i<N; i++)
    {
        printf("  Entra enter (>0): ");         //Entrada dels dos enters
        scanf("%i", &a[i]);
    }
    mcd = a[0];
    for (i=1; i<N; i++)
        mcd = euclides(mcd, a[i]);              //Entrada dels dos enters
    printf(" --> mcd = %i\n", 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 6.6

La codificació és directa.

Diagrama de flux:

Entrades: els nombres enters des de 1 fins a arribar a un nombre **i** que és el **N**-èssim primer.

Sortides: els **N** primers nombres primers, emmagatzemats en l'array **pr[ ]**

Variables de treball. Apart de i, N i pr[ ],

n: enter que porta el compta dels nombres primers ja trobats

primer: variable booleana de treball. És 1 mentre no es troba divisor del nombre i amb què es treballa. Es posa a 0 quan es detecta que no és primer.

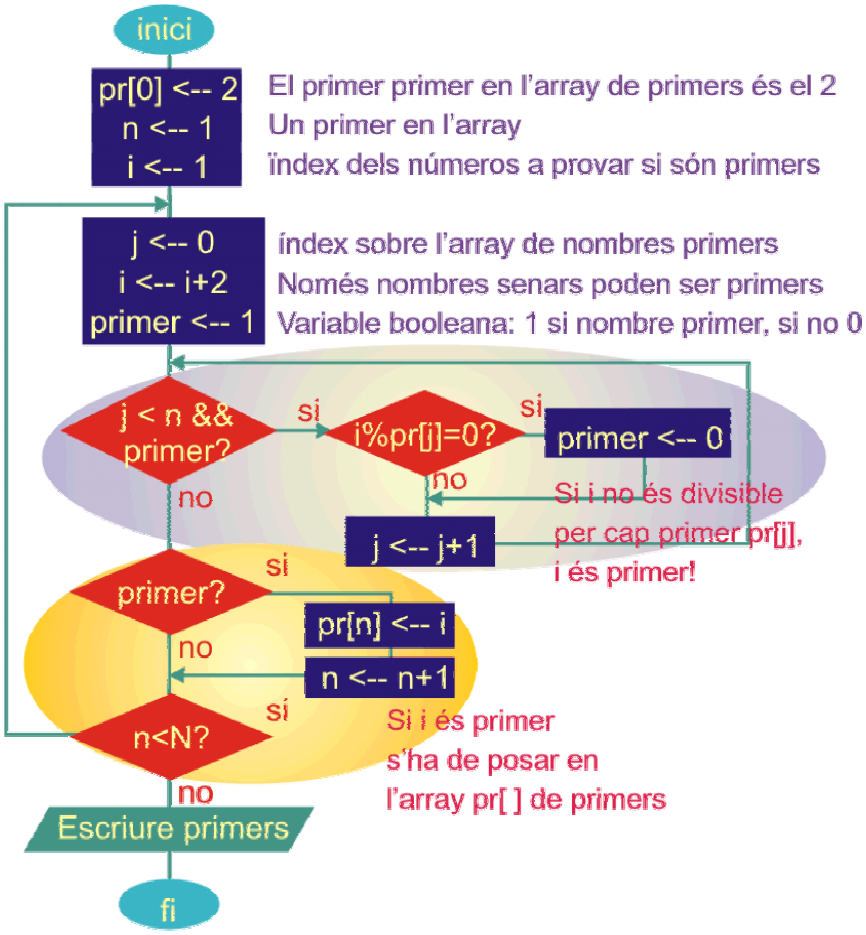
j: índex que passa per tots els primers emmagatzemats per comprovar si el nombre i és primer. Quan més aviat es detecti que un nombre no és primer, més ràpid esdevé l'algorisme. Per a això es comença a comparar amb els enters primers més petits, ja que tenen més múltiples (estem parlant de nombres finits) que els nombres majors.

S'observa que només es tenen en compta els nombres senars (amb i ← i+2) → Cap parell, excepte el 2, és primer!

El programa consta de 4 parts principals:

- Inicialització.
- Detecció si el nombre i és primer
- Si el nombre i és primer, emmagatzematge en pr[ ].
- Impressió dels nombres primers.

El diagrama de flux és el següent:



Codificació en C

```c
/*
Càlcul dels N nombres primers i emmagatzement en un array de N posicions.
*/
#include <stdio.h>
#define N 100
void main()
{
	int i, j;		//Índex nombres naturals senars a comprovar si són primers, índex primers
	int n, pr[N];	//Número de nombres primers trobats, array dels N nombres primers
	int primer;	// Variable booleana: val 1 a menys fins que es demostra que i no és primer
	pr[0]= 2;		//L'únic nombre parell primer és el 2!
	n=1;		// ...en conseqüència, ja tenim un primer!!
	i = 1;
	do
	{
		j=0;
		i += 2;				//Només es comproven els nombres senars
		primer=1;
		while (j<n && primer)		//Si i no és primer, és divisible per un primer
		{
			if (i%pr[j]==0) primer=0;	//Si i no és primer, primer ← 0
			j++;
		}
		if (primer)			//Si és primer s'emmagatzema en pr[ ]
		{
			pr[n]=i;
			n++;
		}
	}while(n<N);
	for (i=0;i<N;i++)			//Imprimir els nombres primers
	{
		printf("%4i", pr[i]);
		if (i%8==0 && i!=0) printf("\n");
	}
	printf("\n");
}
```

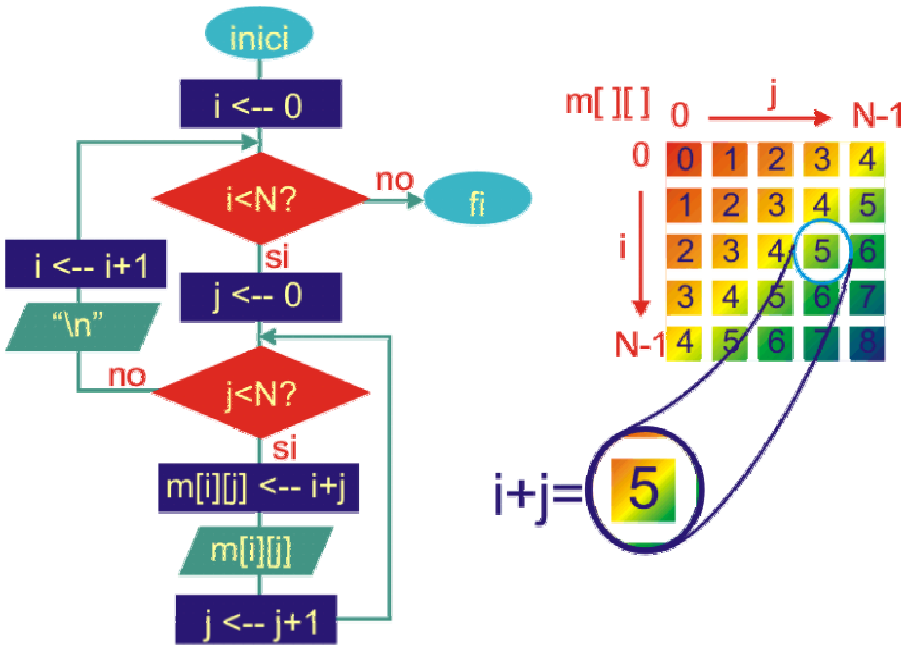
Exercici 6.7

Exercici 6.7a: Sobre diagonals dreta-esquerra:

És un problema senzill de tractament amb índex d'arrays.

El tractament de la diagonal que va (avall) de dreta a esquerra és immediat: es compleix que i+j=constant!. Com a comprovació a cada diagonal s'hi posarà el valor i+j.

Diagrama de flux:



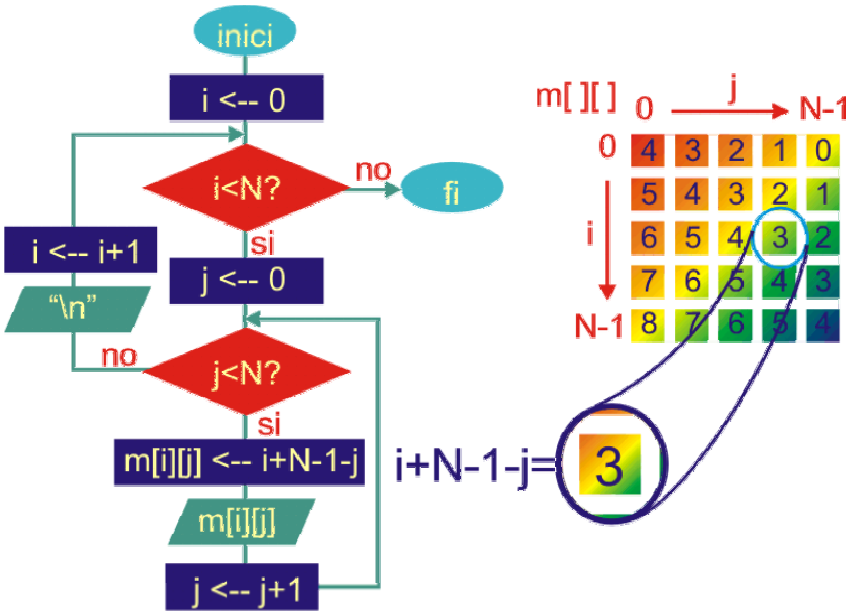
Codificació en C

```
/*
Creació d'una matriu amb valors en la diagonal dreta-esquerra amb el mateix valor.
*/
#include <stdio.h>
#define N 8

void main()
{
	int i, j;				//Índexs per recórrer la matriu
	int m[N][N];				//Matriu N x N cel·les

	printf("MATRIU AMB DIAGONALS IGUALS\n\n");
	for (i=0; i<N; i++)
	{
		for (j=0; j<N; j++)
		{
			m[i][j] = i+j;		//Càlcul cel·la matriu
			printf("%2i ", m[i][j]);	//Imprimir cel·la matriu
		}
		printf("\n");			//Salt de línia per a nova fila de la matriu
	}
}
```

Exercici 6.7b: Sobre diagonals esquerra-dreta:

En aquest cas convé veure que seria interessant fer el compteig de les j al revés de com es fa (de dreta a esquerra), i ens trobaríem amb el mateix algorisme anterior. Això és equivalent a posar en cada cel·la: i+(N-1-j)

Diagrama de flux:



Codificació en C

```
/*
Creació d'una matriu amb valors en la diagonal esquerra-dreta amb el mateix valor.
*/
#include <stdio.h>
#define N 8

void main()
{
	int i, j;				//Índexs per recórrer la matriu
	int m[N][N];				//Matriu N x N cel·les

	printf("MATRIU AMB DIAGONALS IGUALS\n\n");
	for (i=0; i<N; i++)
	{
		for (j=0; j<N; j++)
		{
			m[i][j] = i+(N-1)-j;		//Càlcul cel·la matriu
			printf("%2i ", m[i][j]);	//Imprimir cel·la matriu
		}
		printf("\n");				//Salt de línia per a nova fila de la matriu
	}
}
```

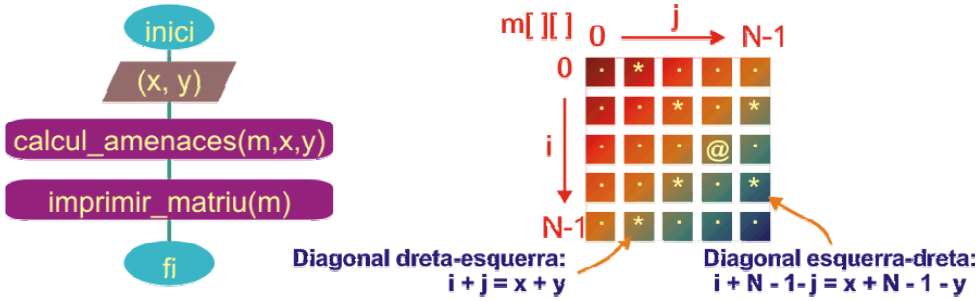
Exercici 6.7c: Parlant sobre àlfils...

Les caselles amenaçades en el tauler d'escacs són immediates d'obtenir ajuntant els dos algorismes anteriors!

Així, el programa queda reduït a demanar la posició inicial de l'àlfil i a efectuar els càlculs de les cel·les amenaçades (ja fet en els dos apartats anteriors). Només es dóna, per tant, el diagrama de flux del programa principal, que crida a la funció càlcul_amenaces() i imprimir_matriu().

Diagrama de flux:

El diagrama de flux de la funció main() i les condicions a complir en diagonals esquerra-dreta i dreta-esquerra són els següents:



Codificació en C

```
/* Amenaces àlfil!  */
#include <stdio.h>
#define N 8
void calcul_amenaces(char m[N][N], int, int);          //Rutina càlcul cel·les amenaçades
void imprimir_matriu(char m[N][N]);                    //Rutina impressió matriu
void main()
{
        int x, y;
        char m[N][N];                                  //Declaració tauler
        printf("AMENACES ALFIL\n");
        printf("Cel.la on es troba l'alfil (x,y) (0<x,y<7): ");   //Entrada posició àlfil sobre tauler
        scanf("%i%i",&x, &y);
        calcul_amenaces(m, x, y);                      //Càlculs amenaces
        imprimir_matriu(m);                            //Impressió matriu
}
void calcul_amenaces(char m[N][N], int x, int y)       //Càlcul cel·les amenaçades
{
        for (int i=0; i<N; i++)
                for (int j=0; j<N; j++)
                {
                        if (x+y == i+j) m[i][j]='*';           //Diagonal dreta-esquerra → *
                        else if (x+N-1-y == i+N-1-j) m[i][j]='*'; // Diagonal esquerra-dreta → *
                        else m[i][j]=0xfa;                     //Cel·la no amenaçada → ·
                }
        m[x][y]='@';                                   //Cel·la on està l'àlfil → @
}
void imprimir_matriu(char m[N][N])                     //Rutina impressió matriu
{
        for (int i=0; i<N; i++)
        {
                for (int j=0; j<N; j++)
                        printf("%c ", m[i][j]);
                printf("\n");
        }
}
```

Exercici 6.8

Exercici 6.8a: Permutacions amb repetició.

És un exercici simple.

Es tracta d'un triple bucle per a escriure totes les permutacions possibles que es poden fer amb una paraula de tres lletres.

Pseudocodi.

Entrades: 3 lletres entrades en una paraula

Sortides: Les 3^3 permutacions

```
Const Enter L=3                                          //Número de caràcters a llegir
Enter i, j, k                                            //Índexs
Caracter cadena[L]                                       //Paraula a entrar
Inici
    LLegir (cadena)
    Per (i=0; i<L) Fer
        Per (j=0; j<L) Fer
            Per (k=0; k<L) Fer
                Escriure(cadena[i], cadena[j], cadena[k])
                k ← k+1
            FiPer
            j ← j+1
        FiPer
        i ← i+1
    FiPer
    Escriure("Nombre permutacions =",i*j*k)          //Escriure nombre de permutacions
Fi
```

Codificació:

```
/*
Permutacions possibles amb tres lletres
*/
#include <stdio.h>
#define L 3

void main()
{
    char cadena[L+1];                                    //Tres lletres més '\0'
    int i, j, k;

    printf("---PERMUTACIONS ---\n");
    printf("Entra una paraula de %i lletres (diferents): ", L);      //Entrar paraula
    scanf("%s", &cadena);
    for (i=0; i<L; i++)                                  //Bucle primera lletra
        for (j=0; j<L; j++)                              //Bucle segona lletra
            for (k=0; k<L; k++)                          //Bucle tercera lletra
                printf("%c%c%c ", cadena[i], cadena[j], cadena[k]);
    printf("\n        El nombre de permutacions es de %i\n", i*j*k);
}
```

Exercici 6.8b: Permutacions sense repetició.

Com abans l'algorisme no té complicacions. Només cal controlar que no hi hagi lletres repetides.

Pseudocodi.

Entrades: 3 lletres entrades en una paraula

Sortides: Les 3! permutacions sense repetició

```
Const Enter L=3                                          //Número de caràcters a llegir
Enter i, j, k                                            //Índexs
Caracter cadena[L]                                       //Paraula a entrar
Inici
    LLegir (cadena)
    Per (i=0; i<L) Fer
        Per (j=0; j<L) Fer
            Si (i<>j)  Llavors
                Per (k=0; k<L) Fer
                    Si (i<>k i j<>k) Llavors
                        Escriure(cadena[i], cadena[j], cadena[k])
                    FiSi
                    k ← k+1
                FiPer
            FiSi
            j ← j+1
        FiPer
        i ← i+1
    FiPer
    Escriure("Nombre permutacions =",i*j*k)          //Escriure nombre de permutacions
Fi
```

Codificació:

```
/* Permutacions sense repetició possibles amb tres lletres */
#include <stdio.h>
#define L 3                                              //Número de lletres
void main()
{
    char cadena[L+1];                                    //Tres lletres més '\0'
    int i, j, k;
    printf("--- PERMUTACIONS SENSE REPETICIO ---\n");
    printf("Entra una paraula de %i lletres (diferents): ", L); //Entrar paraula
    scanf("%s", &cadena);
    for (i=0; i<L; i++)
    {
        for (j=0; j<L; j++)                              //Bucle primera lletra
        {
            if (j!=i)                                    //Dues primeres lletres no repetides
                for (k=0; k<L; k++)                      //Bucle segona lletra
                {                                        //Bucle tercera lletra
                    if (i!=k && j!=k)                    //Dues darreres lletres no repetides
                        printf("%c%c%c ",cadena[i],cadena[j],cadena[k]);
                }
        }
    }
}
```

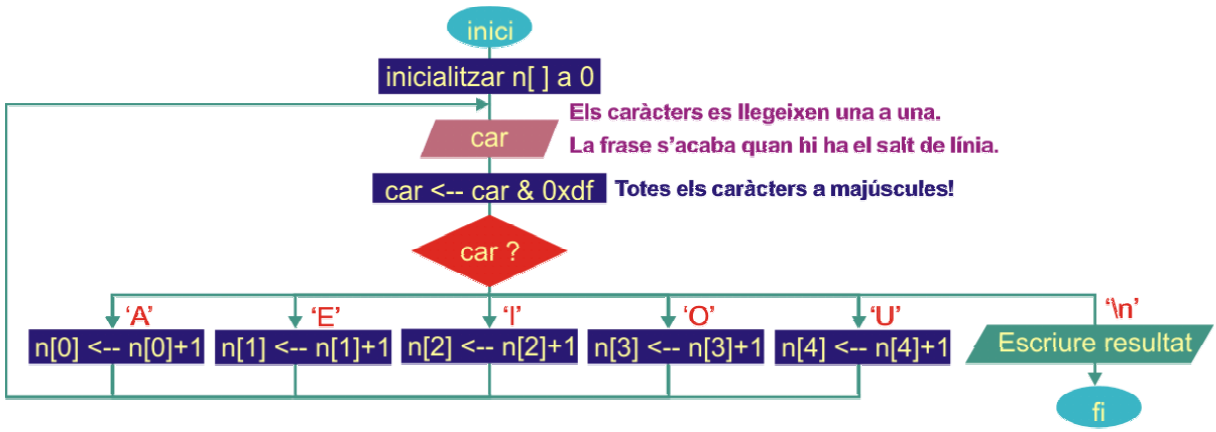
Exercici 6.9

L'algorisme és directe. Conforme s'entra la frase per teclat els caràcters es passen a majúscules i es mira si són vocal.

Entrades: frase

Sortides: Compteig del nombre de vocals entrat

Diagrama de flux:

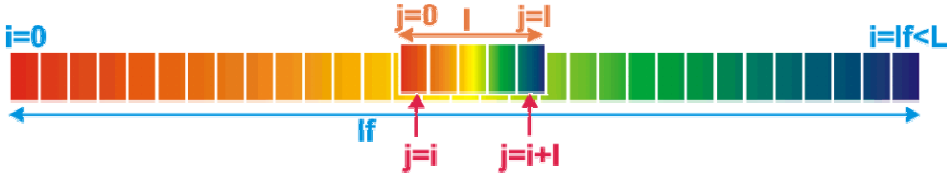


Codificació en C

```
/*
Comptar els cops que cada vocal apareix repetida en una frase
*/
#include <stdio.h>
void main()
{
	int i, n[5];				//Array de comteig: 0->A, 1->E, 2->I, 3->O, 4->U
	char car;
	for(i=0;i<5;i++) n[i]=0;			//Inicialització
	printf("Entra frase: ");			//Entrar frase
	fflush(stdin);
	do
	{
		car=getchar() & 0xdf;			//Tractament per caràcter: tots a majúscules
		switch(car)
		{
		case 'A': n[0]++;
			break;
		case 'E': n[1]++;
			break;
		case 'I': n[2]++;
			break;
		case 'O': n[3]++;
			break;
		case 'U': n[4]++;
		}
	}while(car!='\n');
	printf("\n Aparicio de cada vocal:\n");		//Impressió
	printf("\ta\te\ti\to\tu\n");
	for (i=0; i<5; i++) printf("\t%1i",n[i]);
	printf("\n");
}
```

Exercici 6.10

L'algorisme crea una finestra de comparació de l caràcters (longitud de la subfrase) amb què es va comparant amb la frase principal (de longitud lf).



Els passos principals de l'algorisme són:

i) Entrar frase.

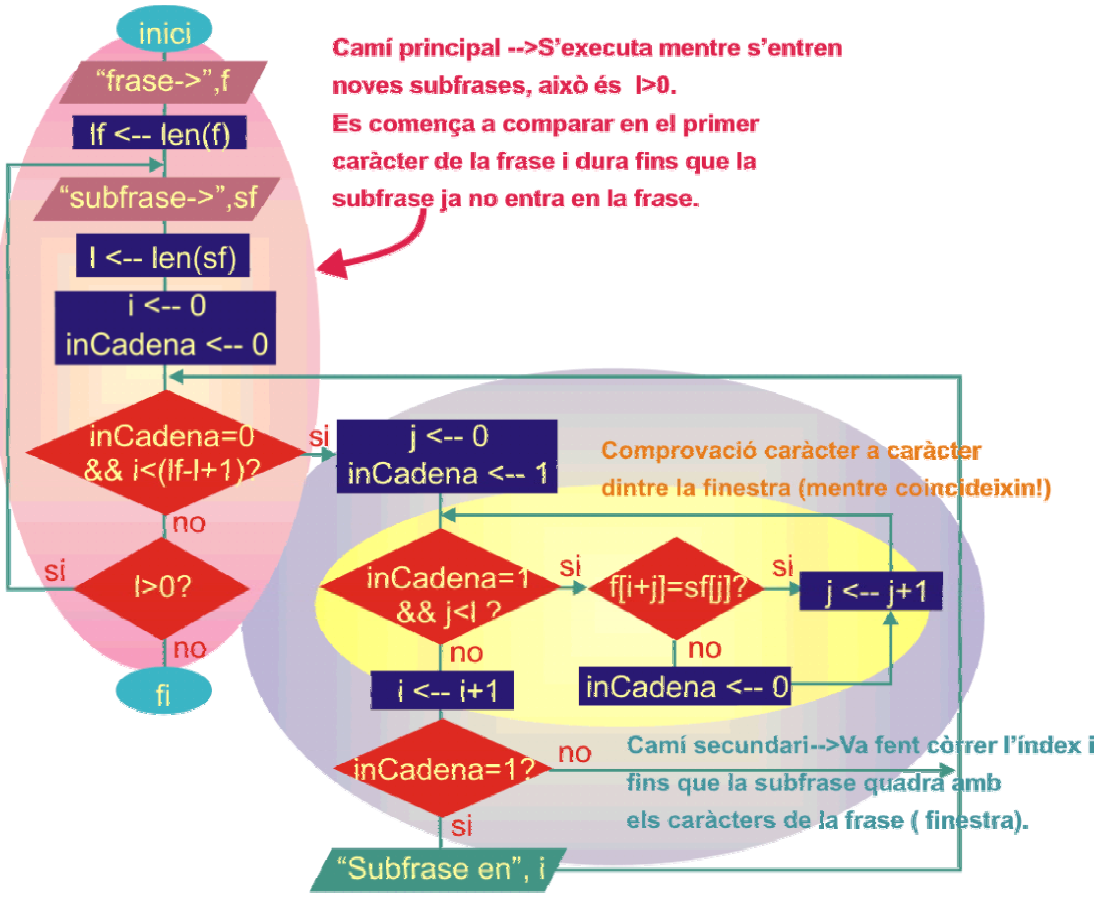
ii) Entrar subfrase. Si la subfrase té algun caràcter s'entra a comparar amb la frase. Si no s'acaba.

iii) Es recorre la frase amb un index i que va des de 0 fins a (lf-l). Quan es detecta que els caràcters de la finestra de comparació no coincideixin amb la subfrase, s'incrementa l'índex i.

iv) Quan la subfrase coincideix amb els caràcters de la finestra (emprant un índex j que va des de j=i fins a j=i+1) s'acaba l'algorisme. Es dóna la posició en la que comença la subfrase dintre la frase.

El diagrama de flux en detalla el funcionament.

Diagrama de flux:



```c
/*
Subcadena en cadena
*/

#include <stdio.h>
#include <string.h>
#define L 80                        //Màxima longitud de la cadena

void main()
{
        char f[L], sf[L];           //Variables emmagatzement frase i subfrase
        int i, j;                   //Índexs
        int lf, l, inCadena;        //Longitud frase, subfrase, variable booleana subfrase trobada

        printf("-- SUBCADENA EN CADENA --\n");
        printf("Entra una frase: ");
        fflush(stdin);
        gets(f);
        lf=strlen(f);                                   //Longitud subfrase

        do
        {
                printf("Entra subfrase a cercar: ");    //Subfrase a cercar
                fflush(stdin);
                gets(sf);
                l=strlen(sf);                           //Longitud subfrase
                i=0;
                inCadena=0;

                while (!inCadena && i<lf-l+1)           //Comença la cerca des d'inici de frase
                {
                        j=0;
                        inCadena=1;                     //Se suposa subfrase en frase

                        while (inCadena && j<l)         //Mentres subfrase en frase inCadena=1
                        {
                                inCadena = (f[i+j]==sf[j]);
                                j++;
                        }               //Si se surt amb inCadena=1 --> subfrase en frase

                        i++;
                        if (inCadena) printf("Subfrase en posicio %i\n", i);
                }                       //Se surt del bucle amb primera aparició de la subfrase

        }while (l>0);
}
```

Exercici 6.11

És un clàssic exercici de treball amb matrius.

Pseudocodi:

Programa principal

Entrades: La matriu 3 x 3
Sortides: El determinant de la matriu
El programa principal no fa res més que cridar a accions:

```
Real m[3][3]                    //la matriu
Real det                        //el determinant
Inici
        entrarMatriu(m);        //Procediment que demana els valors de la matriu
        Per (i=0; i<D)
                det += m[0][i] * det2(m,i); //Càlcul del det(m) a partir dels tres menors d'ordre 2
                i ← i+1
        FiPer
        escriureMatriu(m,det);  //Escriptura del resultat
Fi
```

Procediments entrarMatriu ( ) i escriure Matriu( )

No tenen cap secret: són dos bucles anidats per entrar (escriure) tots els termes (i, j) de la matriu.

Funció det2(m, i)

Entrades: la matriu i l'índex del menor a calcular (mitjançant l'enter i)

Sortides: el determinant del menor

```
Real m[3][3]                    //la matriu
Enter i                         //índex del menor
Inici
        m[1][(i+1)%D]*m[2][(i+2)%D]-m[1][(i+2)%D]*m[2][(i+1)%D]      //Càlcul |menor(i)|
        Retornar(d);                                                //Retorn resultat
Fi
```

Codificació en C

```
/*
Càlcul del determinant d'una matriu de 3x3 a partir dels determinants dels seus menors
*/

#include <stdio.h>
#define D 3                                             //Dimensió de la matriu

void entrarMatriu (float [D][D]);                       //Declaració funció entrar
float det2(float[D][D] , int);                          //Càlcul determinant d'ordre 2
void escriureMatriu (float [D][D], float);              //Declaració funció escriure resultat

void main()                                             //Programa principal
{
        float det = 0;
        float m[D][D];
        entrarMatriu(m);
        for (int i=0; i<D; i++)   det += m[0][i] * det2(m,i);     //Càlcul det(m) cridant a det2(menors m)
        escriureMatriu(m,det);
}
```

```
void entrarMatriu (float m[D][D])                    //Definició funció entrar
{
        int i, j;
        for (i=0; i<D; i++)
        {
                printf("Entra numeros fila %i: ", i);
                for (j=0; j<D; j++) scanf("%f", &m[i][j]);
        }
}

float det2(float m[D][D], int i)                     //Càlcul determinant d'ordre 2
{
        return(m[1][(i+1)%D]*m[2][(i+2)%D]-m[1][(i+2)%D]*m[2][(i+1)%D]);
}


void escriureMatriu (float m[D][D], float dt)        //Escriu en format |m| = dt
{
        printf("\n");
        for (int i=0; i<D;i++)
        {
                printf(" | ");
                for (int j=0; j<D; j++)   printf("%3.1f ", m[i][j]);
                printf("|");
                if (i==D/2) printf(" = %f", dt);
                printf("\n");
        }
}
```

Exercici 6.12

Exercici simple que combina amb elegància el treballar amb índexs.

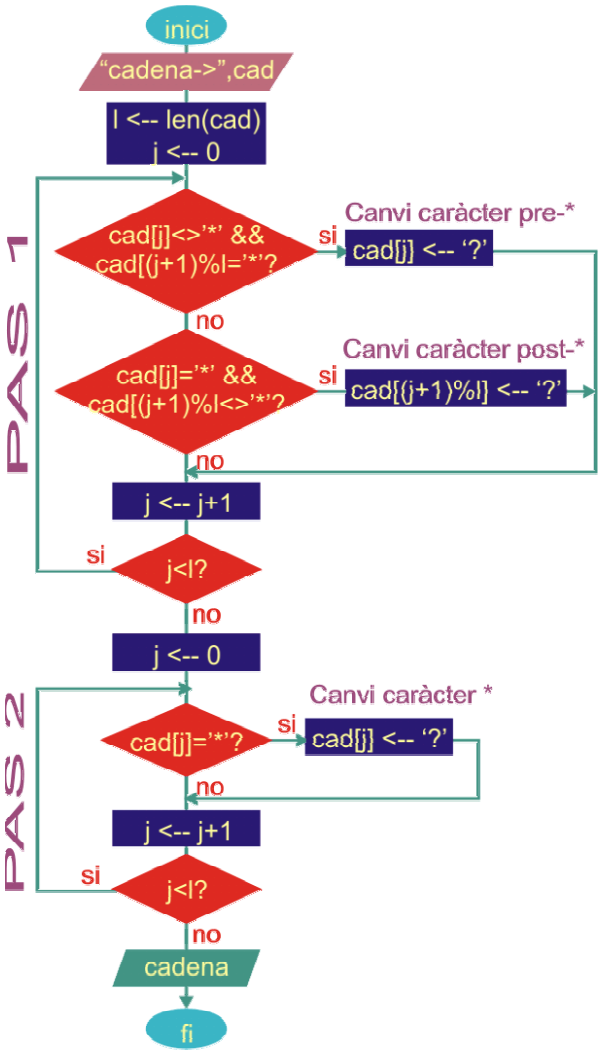
Apart de l'entrada de la cadena i la sortida de la nova cadena, la part interessant de l'exercici és la part de procés on, realment, és on es treballa amb índexs.

Per a substituir els caràcters l'algorisme treballa en dos passos: primer es canvien els caràcters anterior i posterior als * i, en una segona passada, es canvien els *.

Codificació en C

```
/*
Canvi caràcters ...X*X... per ...?@?... en una cadena circular, essent X qualsevol caràcter
*/
#include <stdio.h>
#include <string.h>

const int L=80;    //Longitud màxima de la cadena

void main()
{
        int j, l;                   //Índex, longitud cadena
        char cad[L];                //Array cadena
//Entrar la cadena
        printf("\t-- LLISTA CIRCULAR --\n");
        printf("\t Entra la cadena   --> ");
        gets(cad);
        l= strlen(cad); //Longitud de la cadena
//Pas 1: detecció caràcters anterior i posterior a *
        j=0;
        for (j=0; j<l; j++)
        {
            if (cad[j]!='*' && cad[(j+1)%l]=='*')
                cad[j]='?';          //Caràcter anterior
            else if (cad[j]=='*' && cad[(j+1)%l]!='*')
                cad[(j+1)%l]='?'; //Caràcter posterior
        }
//Pas 2: detecció *
        for (j=0; j<l; j++) if (cad[j]=='*') cad[j]='@';
//Escriptura cadena final
        printf("\n\tLa cadena final es --> %s\n", cad);
 }
```

Diagrama de flux:



Exercici 6.13

HISTOGRAMA DE NOMBRES ALEATORIS

La generació de nombres aleatoris es fa emprant la funció rand() de la llibreria stdlib.h que dóna una sortida float.

La sortida està limitada per la constant RAND_MAX. Per tant, dividint la sortida per aquesta constant es normalitza entre 0 i 1.

Per evitar que la sortida es repeteixi en diferents inicis del programa, s'empra una llavor inicial generada amb funció srand(time(NULL)). Aquesta variable temporal permet variar la condició inicial generadora. Com a conseqüència, s'ha d'observar que també s'ha de declarar la llibreria time.h

Pseudocodi:

Programa principal

```
Entrades: Cap
Sortides: L'histograma
Const Enter NMAX=50
Enter h[NMAX]                          //array de dades
Enter i, n, a                          //index, número de dades, variable de compteig
Inici
        GenerarLlavor                  //Amb srand(time(NULL))
        Llegir(numero_dades)
        Per (i=0; i<numero_dades)
                a=Enter(NMAX*aleatori) //aleatori ← rand()/RAND_MAX
                h[a] ← h[a]+1
                i ← i+1
        FiPer
        Imprimir_histograma
Fi
```

Imprimir_histograma

```
Const Enter NMAX=50
Enter h[NMAX]                          //array de dades
Enter i, j, num, max=0                 //indexs, número amb més dades, freq.màxima
Inici
        Escriure (eix escala)          //Imprimir eix d'escala
        Per (i=0; i<NMAX)              //Cercar freqüència màxima d'aparició
                Si h[i]>max Llavors
                        max=h[i]
                        num=i
                FiSi
                i ← i+1
        FiPer
        Per (i=0; i<=max)              //Impressió histograma
                Per (j=0; j<NMAX)
                        Si h[j]>max Llavors Escriure("*")
                        SiNo Escriure(" ")
                        FiSi
                        j ← j+1
                FiPer
                Escriure (salt de línia)
                i ← i+1
        FiPer
        Escriure ("Freq.Max.=",max , " a",num)
Fi
```

Codificació.

```
/*
HISTOGRAMA DE NOMBRES ALEATORIS
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define NMAX 50

int h[NMAX];                                            //Array de nombres aleatoris
void imprimir(void);
void main()
{
        int i, n, a;
        printf("\nGENERADOR DE NOMBRES ALEATORIS\n\n");
        printf("\nHistograma de n nombres aleatoris entre 0 i %i\n", NMAX);
        printf("...quants nombres vols? ");
        scanf("%i", &n);
        srand(time(NULL));                              //Generador llavor
        for (i=0;i<n;i++)
        {
                a=int(NMAX*rand()/RAND_MAX);            //Generació números aleatoris
                h[a]++;
        }
        imprimir();
}
void imprimir(void)                                     //Rutina impressió números aleatoris
{
        int i, j, num, max=0;
        //Impressió eix d'escala
        for (i=0;i<=NMAX; i++)          if (i%5==0) printf("%2i    ", i);
                printf("\n");
        for (i=0;i<=NMAX; i++)          if (i%5==0) printf("%c", 192);
                                                    else printf("%c", 196);
        printf("\n");
        //Cercar el màxim
        for (i=0; i<NMAX;i++) if (h[i]>max)
                                                    {
                                                            max=h[i];
                                                            num=i;
                                                    }
        //Imprimir histograma
        for (i=0;i<=max;i++)
        {
                for(j=0;j<NMAX;j++) if(h[j]>i) printf("*");
                                                    else printf(" ");
                printf("\n");
        }
        printf("\nMax.freq. = %i, a %i\n\n", max, num);
}
```

Exercici 6.14

És un exercici molt complert que permet treballar plenament amb les operacions que normalment es realitzen amb un array d'enters ordenats de menor a major

L'array de N enters des del conjunt buit (n=0), i es realitzen, amb ell, les operacions de cercar enter, eliminar enter, insertar enter (sempre i quan no existeixi), visualitzar l'array i sortir.

Codificació en C

```
/* Array de N enters creat des del conjunt buit (n=0) amb els elements ordenats de menor a major.
El programa simplement no és res més que un distribuïdor de les operacions a realitzar:
        Cercar enter
        Eliminar enter
        Insertar enter (sempre i quan no existeixi)
        Veure l'array
        Sortir
*/
//Declaració de llibreries
#include <stdio.h>
#define N 10                                            //Longitud de l'array

//Declaració de procediments
void cercar (int [], int, int);                         //Declaració funció cercar
int insertar (int [], int);                             //Declaració funció insertar
int eliminar (int [], int);                             //Declaració funció eliminar
void veure (int [], int);                               //Declaració funció eliminar

//Programa principal
void main()
{
        char c;                                 //Caràcter de decisió de tasca a realitzar
        int ll[N], n=0, num;                    //Array d'enters, número nombres entrats i número
        printf("\t--- ARRAY D'ENTERS ---\n");
        do
        {
                printf("\n\t...(c)ercar, (i)nsertar, (e)liminar, (v)eure, (s)ortir? ");
                fflush(stdin);
                scanf("%c", &c);
                switch(c)
                {
                case 'c': if (n==0)  printf("\t\t --> ARRAY BUIT!\n");          //Cas cercar
                        else
                        {
                                printf("\t\t --> Entra el numero a cercar: ");
                                scanf("%i", &num);
                                cercar (ll, n, num);
                        }
                        break;
                case 'i': if (n<N)          n = insertar (ll, n);               //Cas insertar
                        else printf("\t\t --> ARRAY PLE\n");
                        break;
                case 'e': if (n==0)  printf("\t\t --> ARRAY BUIT!\n");          //Cas eliminar
                        else n = eliminar (ll, n);
                        break;
                case 'v': veure (ll, n);                                        //Cas visualitzar
                }
        }while (c!='s');                                                        //Si no, se surt!
}
```

```
// Definició funció cercar.  Paràmetres passats: array, tamany array, número a cercar
void cercar (int a[N], int n, int x)
 {
        int i=0;
        while (i<n && a[i]!=x) i++;
        if (i==n) printf("\t\t --> Enter no existent!\n");
        else printf("\t\t --> Enter en posicio %i\n", i);
}

//Definició funció insertar.  Paràmetres passats: array, tamany array Retorna: tamany array
int insertar (int a[N], int n)
{
        int i, j, x;
        printf("\t\tEntra numero: ");
        scanf("%i", &x);                                        //Numero a insertar
        i=0;
        while (i<n && a[i]<x) i++;
        if (a[i]==x) printf("\t\t --> Enter ja en posicio %i\n", i);
        else
        {
                for (j=n; j>i; j--) a[j]=a[j-1];
                a[i]=x;
                n++;
                printf("\t\t --> Enter insertat en posicio %i de %i enters\n", i, n);
        }
        return(n);
}

//Definició funció eliminar.  Paràmetres passats: array, tamany array. Retorna: tamany array
int eliminar (int a[N], int n)
{
        int i, j, x;
        printf("\t\tEntra numero: ");
        scanf("%i", &x);                                        //Numero a eliminar
        i=0;
        while (i<n && a[i]<x) i++;
        if (a[i]!=x) printf("\t\t --> Enter no existent!\n", i);
        else
        {
                for (j=i; j<n; j++) a[j]=a[j+1];
                n--;
                printf("\t\t --> Queden %i enters en l'array!\n", n);
        }
        return(n);
}

//Definició funció visualitzar array.  Paràmetres passats: array, tamany de l'array
void veure (int a[N], int n)
{
        for (int i=0; i<n; i++) printf(" %i,", a[i]);
        if (i==0) printf ("\t\t --> ARRAY BUIT\n");
        else  if (i==N) printf ("\n\t\t --> ARRAY PLE\n");
}
```

Exercici 6.15

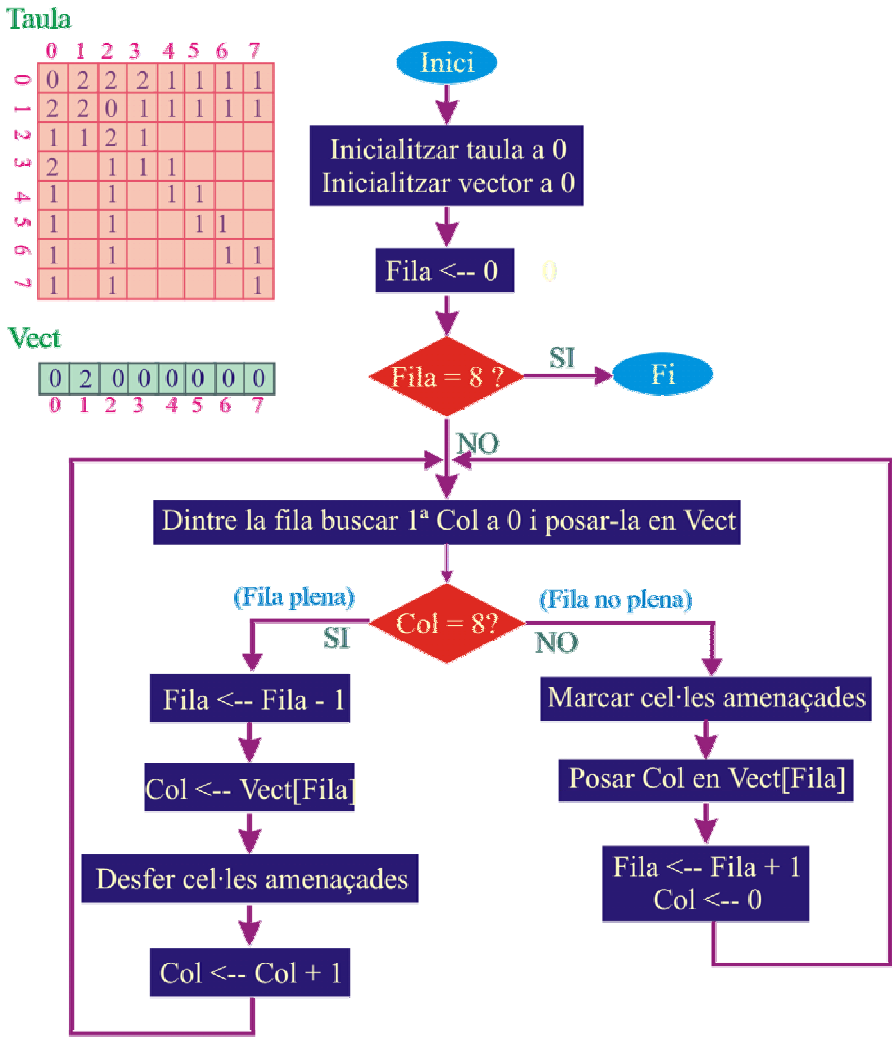
El procés principal el realitza el programa principal. El flux que es porta a terme és el següent:

Variables:
n → número de cel·les per fila/columna
posX, posY → cel·les del tauler on es posen reines
vect[posX] → Guarda posició de la columna on es posa la reina en cada fila

Algorisme:

```
posX ← 0, posY ← 0 //(reina al començament, tot i que el programa permet triar columna inicial)
Mentre (posX<n)        //Encara queden reines per posar
        Cercar primer posY no amenaçat
        Si (posY<n) Llavors
                Incrementar amenaces en cel·les que amenaça la reina
                vect[posX] ← posY     //Vector que guarda posY per cada posX
                posX ← posX+1
                posY ← 0
        SiNo
                posX ← posX-1         //Anar a fila anterior
                posY ← vect[posX]     //Recuperar posY de la reina posada en fila anterior per...
                Decrementar amenaces en cel·les que amenaça la reina
                posY ← posY+1         //Provar nova columna
        FiSi
FiMentre
```

Diagrama de flux del procediment principal:



Codificació

```
/*
Posicionament n reines en un tauler n*n (n>3)
*/
#include <stdio.h>
#include <time.h>
#define n 8                                     //Numero de reines, n>3
#define forma 0                                 //Forma d'impressió

int tauler[n][n];                               //El tauler d'escacs
int vect[n];                    //Guardarà l'última columna visitada de la fila. Útil en els backtracks

//Declaració de procediments
int Inicialitzacions(void);
void ImprimirTauler(double, int);
void FerDesferAmenaces(int, int, int);
int CercarPrimerZero(int, int);

//Programa principal
void main()
{
        int posX=0,posY=0;              //posX, posY --> Posició on es prova de posar la reina actual
        double numPassos=0;             //variable de càlcul de backtracks que es fan en el programa
        clock_t total;                  //Variables per càlcul del temps de procés
        double temps;                   //Guarda el temps total de procés

        printf("Posicionament de %i reines en un tauler %i*%i\n", n, n, n);
        posY = Inicialitzacions();

        total=clock();                  //Comença el procés
        while (posX<n)                  // Les reines estan posades quan n'hi ha una en cada fila
        {
                posY=CercarPrimerZero(posX, posY);      //Cerca primera columna buida
                if (posY<n)             //Si hi ha columna es calculen noves cel·les amenaçades
                {
                        FerDesferAmenaces(posX, posY, 1);       //1 ==> posar amenaces
                        vect[posX]=posY;        //Guardar la columna on s''ha posat la reina
                        posX++;                 //Incrementar fila
                        posY=0;                 //Anar a columna 0 en nova fila
                }
                else                    //No hi ha columna lliure en fila, s'han de desfer passos fets
                {
                        posX--;                 //Anar a fila anterior
                        posY=vect[posX];        //Agafar columna on s'havia posat la reina
                        FerDesferAmenaces(posX, posY, -1);      // -1 ==> desfer les amenaces
                        posY++;                 //Anar a nova columna en la fila anterior
                }
                numPassos++;
                if (!((int)numPassos%1000)) printf("%6.0f\r",numPassos); //Només per seguiment
        }
        temps = (double)(clock() - total) / CLOCKS_PER_SEC;     //Temps de procés
        printf("Temps de proces = %3.2f", temps);
        ImprimirTauler(numPassos, forma);                       //Imprimir posicionament reines
}
```

```
//Inicialitació: Inicialitza array tauler. S'entra la columna de la primera reina
int Inicialitzacions(void)
{
	int i, j, posColumna;
	do
	{
		printf(" Dona columna inicial (col<%1i): ", n);
		scanf("%i", &posColumna);
	}while (posColumna<0 || posColumna>=n);
	for (i=0;i<n;i++) for (j=0;j<n;j++) tauler[i][j]=0;		//Inicialitzar tauler
	return(posColumna);
}
/*Imprimeix el tauler i forma d'impressió:
	form=0 --> es veu millor posicionament reines sobre tauler
	form=1 --> s'imprimeix el número d'amenaces que té cada cel·la*/
void ImprimirTauler(double numPas, int form)
{
	int i, j;
	printf("\n");
	for (i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			if (form) printf("%2i",tauler[i][j]);
			else if (tauler[i][j]) printf("%c ",0xfa);
				else printf("* ");;
		printf("\n");
	}
	printf("\tNumero de passos = %2.0f\n\n", numPas);
}
/*Rutina tractament d'amenaces: x, y donen la posició de la reina
	incdec=1 --> es posa reina i es calculen noves cel·les amenaçades
	incdec=0 --> es treu reina i es disminueix el número en les cel·les que amenaçava*/
void FerDesferAmenaces(int x, int y, int incDec)
{
	int i, d, s;
	for (i=0; i<n; i++)					//files i columnes
	{
		tauler[i][y] += incDec;
		tauler[x][i] += incDec;
	}
	s=x+y;							//diagonal inversa
	for (i=0; i<s+1; i++) if (i<n && (s-i)<n) tauler[i][s-i] += incDec;
	d=y-x;							//diagonal directa
	if (d>=0) for (i=0; i<n-d; i++) tauler[i][i+d] += incDec;
	else  for (i=-d; i<n; i++) tauler[i][i+d] += incDec;
	tauler[x][y]=0;						//-->quadre en el que es troba posada la reina
}
/*Quan s'ha posat una reina es busca la columna (de la fila següent) que està lliure
Paràmetres passats: posY a partir d'on es mira si hi ha una columna lliure en la fila posX
Paràmetre retornat: la posY que està lliure (si existeix!)*/
int CercarPrimerZero(int posx, int posy)
{
	while (tauler[posx][posy]!=0 && posy<n) posy++;
	return(posy);
}
```

Solució recursiva...

...algorisme recursiu que resol el problema de les n reines en un taules, aquí solucionat per 8.

```
//Posicionament 8 reines en un tauler 8x8

#include <stdio.h>
#define n 8                                     //Numero de reines

int tauler[n][n];                               //El tauler d'escacs
int vect[n];                                    //Guardarà l'ultima columna visitada de la fila
int posX0, posY0;

void ImprimirTauler(void);
void FerDesferAmenaces(int, int, int);
int CercarPrimerZero(int, int);
void escacs (int, int);                         //paràmetres: PosX, PosY

void main()
{
        escacs(posX0, posY0);
        ImprimirTauler();
}
void escacs (int posX, int posY)
{
        if ((posX-posX0)!=n)
        {
                posY=CercarPrimerZero(posX%n, posY);
                if (posY<n)
                {
                        FerDesferAmenaces(posX%n, posY, 1);     //1 ==> fer
                        vect[posX%n]=posY;
                        escacs(++posX, 0);
                }
                else
                {
                        posY=vect[(--posX)%n];
                        FerDesferAmenaces(posX%n, posY, -1);    // -1 ==> desfer
                        escacs(posX, ++posY);
                }
        }
}
void FerDesferAmenaces(int x, int y, int incDec)
{
        int i, d, s;
        for (i=0; i<n; i++)                                     //files i columnes
        {
                tauler[i][y] += incDec;
                tauler[x][i] += incDec;
        }
        s=x+y;                                                  //diagonal inversa
        for (i=0; i<s+1; i++) if (i<n && (s-i)<n) tauler[i][s-i] += incDec;
        d=y-x;                                                  //diagonal directa
        if (d>=0) for (i=0; i<n-d; i++) tauler[i][i+d] += incDec;
        else  for (i=-d; i<n; i++) tauler[i][i+d] += incDec;
        tauler[x][y]=0;
}
```

```
int CercarPrimerZero(int posx, int posy)
{
	while (tauler[posx][posy]!=0 && posy<n) posy++;
	return(posy);
}
void ImprimirTauler(void)
{
	int i, j;
	printf("\n\t");
	for (i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			if (tauler[i][j]) printf("%c ",0xfa);
			else printf("* ");;
		printf("\n\t");
	}
}
```