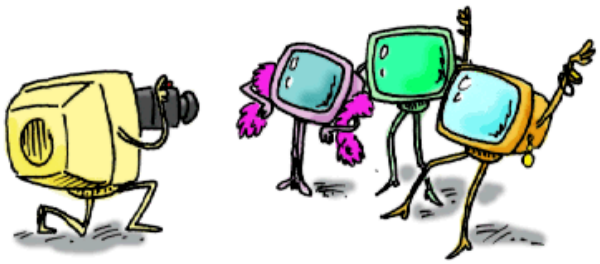
Tema 6: Arrays (llistes i taules)

- 6.1 Introducció
- 6.2. Llistes o vectors
  - Definició.
  - Declaració de llistes
  - Emmagatzament en memòria
  - Tamany de les llistes
  - Inicialització de llistes
  - Exemple
- 6.3 Llistes de caràcters
- 6.4. Matrius
- 6.5 Pas de paràmetres
- 6.6 Operant amb llistes
- 6.7 Exemples



6.1 Introducció

- **Estructura de dades** → Col·lecció de dades que es caracteritzen per la seva organització i operacions definides entre elles.
- Convé recordar que les dades poden ser: :

Simples → La variable representa un element.	Estàndard:	Enteres Reals Caràcter Lògiques
	Definides per l'usuari:	Enumerades Subrang
Estructurades → La variable pot representar múltiples elements.	Estàtiques: L'espai que ocupen queda determinat en el moment de definir-les.	Array (vector/matriu) Cadena (string) Conjunt Registre Fitxer (arxiu)
	Dinàmiques: Ocupen espai en memòria conforme es necessita.	Llista (pila/cua) Arbre Graf

6.2 Llistes o vectors (I)

- **Definició.**
- Array (llista o vector, taula o matriu) → Conjunt finit i ordenat d'elements homogenis. Això és, d'enters, reals, caràcters, registres, ...
- Llista o vector → És un array unidimensional (taula o matriu de dimensió 1).
  - Com que és una estructura ordenada, els seus elements ocupen una posició concreta i poden ser identificats.
  - La numeració de la posició que ocupa cada element en la llista és consecutiva.
  - La llista pot ser referida com a un tot (N), o per cadascun dels seus elements (N(i), i = 1÷100), on cada element és un tipus de dada.
  - Exemple:
  - Sigui Q una llista que conté les notes de 10 alumnes (n = 10):

Q	5.5	9.5	4.5	6.0	8.0	7.5	5.5	4.0	8.5	9.0
i	0	1	2	3	4	5	6	7	8	9

  - Els elements de la llista estan ordenats dels del 0 fins al n-1, on n és el tamany (nombre d'elements de la llista, en aquest cas 10.
  - Q[0] és l'element que hi ha a la posició 0, Q[1] el de la posició 1, ..., on les posicions 0 a 9 es recorren emprant una variable índex.
- **Declaració de llistes.**
- En la declaració de la llista s'ha d'especificar el tipus de variable de les dades.
  - En l'exemple anterior la declaració seria: float Q[]; o float Q[10]; i en C se sobreentén que l'índex agafa els valors que van des del 0 fins al 9.

6.2 Llistes o vectors (II)

- **Emmagatzament en memòria**
- Els elements de la llista s'emmagatzemen en memòria en ordre adjacent.
  - En l'exemple anterior, en tractar-se de 10 elements reals, ocuparien 10*4=40 bytes consecutius de memòria.

Q[]=Q[0]	5.5	Adreça x
Q[1]	9.5	Adreça x+1*4
...		
Q[9]	9	Adreça x+9*4

- **Tamany de les llistes**
  - Mitjançant la funció sizeof() es pot saber el tamany dels arrays.
  - Per exemple:
    - m = sizeof[Q] donarà m=10*4 = 40 bytes
    - m = sizeof(Q[5]) donarà m = 4 bytes.
- **Inicialització de llistes**
  - Els arrays es poden inicialitzar dintre la declaració: int t[5] = {10,15,20,18,21};
  - La inicialització en la declaració, permet ometre el nombre d'elements: int v[] = {10,15,20,18,21};
  - C inicialitza les variables globals a 0 (caràcter nul – '\0' – en el cas de llistes de caràcters) (com si fossin variables externes). Però no ho fa en les variables locals (automàtiques).
  - Una solució per inicialitzar les variables locals és considerar-les estàtiques.

6.2 Llistes o vectors (III)

- **Exemple**

```
//Inicialització de vectors
#include <stdio.h>
void main()
{
    int i, v[3];
    for (i=0; i<3; i++)
        printf("v[%1u] = %u.\n ",i, v[i]);
}
```

Resultat: v[0] = -5678
v[1] = 0
v[2] = 34567
→ Inicialitzat a qualsevol valor!

```
// Inicialització de vectors
#include <stdio.h>
int v[3];
void main()
{
    int i;
    for (i=0; i<3; i++)
        printf("v[%1u] = %u.\n ",i, v[i]);
}
```

Resultat: v[0] = 0
v[1] = 0
v[2] = 0
→ Inicialitzat a 0

6.2 Llistes o vectors (IV)

- Exemple
  - Càlcul de la temperatura promig a les 13:00 hores d'un mes d'abril.

```
/*
Calcul temperatura promig
Exemple d'operativitat amb arrays.
*/
Const Enter DIES 30
Enter i
Real T[DIES]
Inici
   Per (i=0;i<DIES)
      Escriure("Dona T[",i,"] ")
      Llegir(T[i])
      prom = prom+T[i]
      i=i+1
   FiPer
   prom=prom/DIES
Fi
```

```
/*
Càlcul temperatura promig
Exemple d'operativitat amb arrays.
*/
#include <stdio.h>
#define DIES 30
void main()
{
   int i;
   float T[DIES], prom=0;
   for (i=0;i<DIES;i++)
   {
      printf("T[%i]=", i);
      scanf("%f",&T[i]);
      prom+=T[i];
   }
   printf("Tprom=%3.1f\n",prom/DIES);
}
```

6.3 Llistes de caràcters (I)

- String o cadena de caràcters → Conjunt finit i ordenat de caràcters emmagatzemats consecutivament.
- Longitud de la cadena → Nombre de caràcters que conté.
  - Exemple: "Hola" → longitud = 4.
- S'han de tractar com a vectors de caràcters.
- **Declaració**: char nom[n] ò char nom[] = 'Hola', amb n el nombre de caràcters.
- S'ha de distingir entre cadena i llista de caràcters. La cadena al final hi inserta un caràcter nul ('\0') que en facilita el seu tractament.
  - Exemple: Es defineix una cadena a[] i un array de caràcters b[]. Així com el final de cadena es detecta controlant el caràcter '\0', no es pot fer el mateix amb l'array de caràcters.

```
//Cadena (amb '\0' es detecta el final)
void main()
{
        int i=0;
        char a[]="Es dimarts.";
        printf("%s\n", a);
        do {
                printf("%c", a[i]);
                i++;
        }while (a[i]!='\0');
        printf("\nNumCars=%i\n",sizeof(a));
}
→   Es dimarts.
    Es dimarts.
    NumCars=12
```

```
//Array de caràcters (el final no es detecte amb '\0').
void main()
{
        int i=0;
        char b[]={'E','s',' ','d','i','m','a', 'r','t','s','.'};
        printf("%s\n", b);
        do {
                printf("%c", b[i]);
                i++;
        }while (b[i]!='\0');
        printf("\nNumCars=%i\n",sizeof(b));
}
→   Es dimarts.ÿ╗   (o similar!)
    Es dimarts.■    (o similar!)
    NumCars=1
```


6.3 Llistes de caràcters (II)

- Treballar amb cadenes de caràcters sol ser dur. Per això cal recordar que hi ha llibreries que ens faciliten molt la tasca de treballar amb cadenes de caràcters. Entre les funcions que hi tenen definides cal recordar:

Llibreria	Funció	Exemple

stdio.h	gets() → Entrar cadena des d'entrada estàndard
	puts() → Escriure cadena a sortida estàndard
		char cad[20];
		puts("Entra nom: ");
		gets(cad);

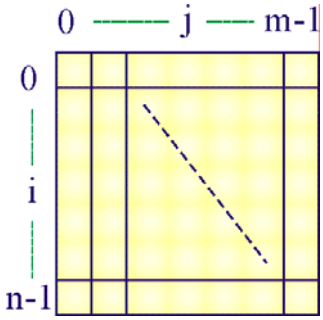
stdlib.h	atoi() → Pas cadena de caràcters a enter
	atof() → Pas ASCII a real
	atol() → Pas ASCII a enter llarg
		char str[8];
		printf("Entra enter amb signe → ");
		gets(str);
		printf("L'enter es %i.\n", atoi(str));

ctype.h	isalnum(), isalpha() → Si és alfanumèric, alfabètic isdigit(), isspace() → Si és dígit, espai islower(), isupper() → Si és minúscula, majúscula	char ch; if (isalpha(ch)!=0) printf("Es alfabètic\n");

string.h	strlen() → Longitud de la cadena
	strcat() → Concatenació de cadenes
	strcomp() → Comparació de cadenes
	strchr() → Cerca un caràcter en la cadena
	strstr() → Cerca una subcadena en una cadena
		char cad1[20]="Can";
		char cad2[]="Formic";
		strcat(cad1, cad2);
		puts(cad1);
		printf("Longitud total = %i",strlen(cad1));

6.4 Taules o matrius (I)

- Matriu → Array bidimensional o multidimensional.
  - S'organitza en files i columnes.
  - **Declaració**: tipus_element nom_taula [<n>][<m>], on n i m són el número de files i el número de columnes.
  - La taula conté n*m elements.
    - Exemple: tauler d'escacs. Es pot definir com una matriu de 8*8 cel·les on en cada cel·la hi pot haver un dels següents elements: 0 → cel·la buida, 1 → peó blanc, 2 → torre blanca, ..., -1 → peó negre, -2 → torre negra, ...
    - → int escacs[8][8];
    - Altres exemples: int Taula [3][2] = {{1,1}, { 10,11}, {20, 21}}
      int codi [3][3] = {0, 1, 2, 3, 4, 5, 6, 7, 8}



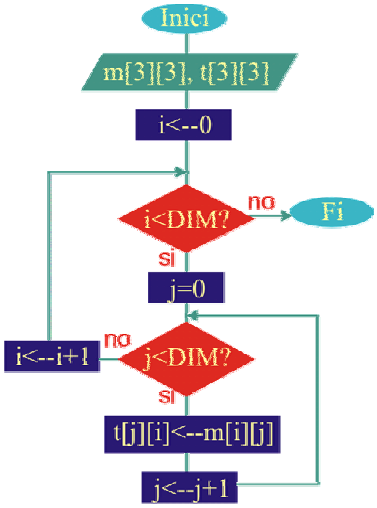
  - **Emmagatzement de la taula en memòria.**
  - És estàtic i consecutiu.
  - En el C s'emmagatzema per files.
  - Això implica que queda ordenat de la forma:
    (0, 0), (0, 1), ...(0, m-1), (1, 0), ...(n-1, m-1)
    on l'element (i, j) ocupa la posició p(i,j) = m*i +j
  - Per tant, l'accés als diferents elements de la matriu requerirà tants bucles for com dimensions tingui la matriu.

6.4 Taules o matrius (II)

- Exemple.

```
/*Càlcul de la matriu trasposta d'una matriu de tres dimensions*/
#include <stdio.h>
#define DIM 3
void imprimir(int h[][DIM], char[]);
void main()
{
      int m[3][3], t[3][3], i, j;                  //Taules inicial i trasposta i índexs per recórrer les taules
      printf("Dona els 9 enters: ");               //Entrar dades
      for (i=0; i<DIM; i++)
        for (j=0; j<DIM; j++)   scanf("%i", &m[i][j]);
      imprimir(m, "Matriu inicial:");              //Imprimir matriu inicial
      for (i=0; i<DIM; i++)                        //Càlcul de la trasposta
        for (j=0; j<DIM; j++)   t[j][i] = m[i][j];
      imprimir(t, "Matriu trasposta:");            //Imprimir trasposta
}
void imprimir(int h[DIM][DIM], char cad[])     //Rutina impressió
{
      printf("\n%s\n\t", cad);
      for (int i=0; i<DIM; i++)
      {
        for (int j=0; j<DIM; j++) printf("%i  ", h[i][j]);
        printf("\n\t");
      }
      printf("\n");
}
```

Algorisme del càlcul de la trasposta



6.5 Pas de paràmetres (I)

- En C tots els arrays es passen per referència, fet que implica que, tota modificació feta en l'array, es mantindrà un cop es torni al programa principal.
- Això és deu al fet que el nom de la variable és una constant que indica la posició de la primera posició de memòria de l'array: &array[]=&array[0].
  - Per tant, quan es crida a una funció que té com a paràmetre un array sembla com si es fés un pas de paràmetres per valor. La funció agafa, però, l'adreça del primer element. En conseqüència, el resultat final és un pas de paràmetres per referència.
  - El següent exercici mostra com, després de la crida de la funció suma(), el vector queda modificat.
- Exemple:

```
#include <stdio.h>
const int DIM=3;
void suma10(int []);            //declaració funció amb array (també acceptat suma10(int[]) ).
void main()
{
    int i, vect[]={3,6,9};                                  //definició llista vect
    for (i=0; i< DIM; i++) printf("iv[%1u] = %u\n", i, vect[i]);   //Impressió llista inicial
    suma10(vect);                                           //Crida funció
    for (i=0; i< DIM; i++) printf("fv[%1u] = %u\n", i, vect[i]);   //Impressió llista final
}
void suma10(int v[])                                        //funció suma10
{
    int i;
    for (i=0; i< DIM; i++) v[i] += 10;
}
```


6.5 Pas de paràmetres (II)

- La generalització del pas de vectors en funcions comporta el problema que la funció té difícil conèixer el darrer element de l'array.
- Una manera d'eliminar aquest problema és passant com a paràmetres tant l'array com el nombre d'elements de què consta.
- Exemple: Pas de minúscules a majúscules

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void aMaj(char[], int);                     //Declaració procediment canvi a majúscules
void main()
{
    char cad1[]="ara ho veureu...";                 //Cadena de prova 1
    char cad2[]="i despres tothom ho entendra!";    //Cadena de prova 1
    aMaj(cad1, strlen(cad1));                       //Crida funció per cadena 1
    aMaj(cad2, strlen(cad2));                       //Crida funció per cadena 2
    puts(cad1);
    puts(cad2);
}
void aMaj(char cad[], int n)                // Definició Procediment canvi a majúscules
{
    for (int i=0; i<n; i++)  if (isalpha(cad[i])) cad[i] -= 0x20;
}
```

6.5 Pas de paràmetres (III)

- El mateix pas es pot fer recollint l'array amb un apuntador (de fet, és el mateix que s'ha fet en l'exemple anterior!):
- Exemple: Funció que compta el número de paraules d'una frase.

```
#include <stdio.h>
#include <string.h>

/*Comptar paraules d'una frase*/

int paraules(char*, int);
void main()
{
      char fr[80];
      int n;
      printf("\tEntra una frase:\t");
      gets(fr);
      n=paraules(fr, strlen(fr)); //es passa una adreça
      printf("\tEn total %i paraules.\n", n);
}
```

```
int paraules(char *f, int l)
{
      int num=0, i=0, espai=1;
      for (i=0;i<l;i++)
      {
          if (f[i]==' ') espai=1;
          else
          {
              if (espai==1)
              {
                  espai=0;
                  num++;
              }
          }
      }
      return(num);
}
```


6.6 Operant amb llistes (I)

- Les llistes proporcionen una eina potent de treball amb dades.
- Per això, hi ha un piló d'operacions que esdevenen estàndard quan es treballa amb arrays.
- Entre aquestes operacions hi ha: recórrer la llista, cercar un element de la llista, afegir elements, insertar elements (enmig la llista), esborrar elements i ordenar els elements de la llista.
- La realització d'aquestes funcions és molt simple, tal com es veu a continuació.

- Sigui un array A de N elements amb m plens (de l'element 0 fins al m-1 (<n)).

- Funció que cerca un element x de la llista:

```
i ← 0
trobat ← fals
Mentre ((!trobat) i (i < m)) Fer
     Si A[i] = 'x' Llavors trobat = cert    //x → dada a cercar
     SiNo Llavors i = i+1
FiMentre
Si trobat ...
```

6.6 Operant amb llistes (II)

- Funció que afegeix un element en la llista:
  - Inserció al final
  - El darrer element passarà a ser l'element m
  - Per tant, s'ha d'incrementar m

```
Si m = N Llavors Escriure("Array ple")
SiNo Llavors
    A[m] ← x            //x = dada a posar
    m ← m+1
FiSi
```

- Funció que inserta un element en la posició i de la llista:
  - Tots els elements des de j fins a m han de córrer des de la posició j+1 a la m+1

```
Si m = N Llavors Escriure("Array ple")
SiNo Llavors
    Per (i=m; i>=j) Fer    //Córrer posicions per fer espai
        A[i] ← A[i-1]
        i ← i-1
    FiPer
    A[i] ← A[j]            //A[j] és l'element a insertar
    m ← m+1
FiSi
```


6.6 Operant amb llistes (III)

- Funció que esborra l'element i (<m):
  - Només cal fer córrer els elements

```
Si i >= m Llavors Escriure ("No té sentit")
SiNo Llavors
    Per (j=i; j<m) Fer
        A[j] ← A[j+1]      //S'escriu (s'esborra) l'element A[j]
        j ← j+1
    FiPer
    m ← m-1
FiSi
```

- Algorisme d'ordenació de la bombolla. L'algorisme de la bombolla probablement és l'algorisme d'ordenació més conegut.

  Sigui un array de n elements a ordenar. Els passos són (ordenació de menor a major):

  1. Començant des de l'element 0 i anant fins al n-1 es comparen els elements 2 a 2. Si el primer element (dels 2 que es comparen) és més gran que el segon, s'intercanvien. Si no, no es fa res i es passa a la següent comparació.

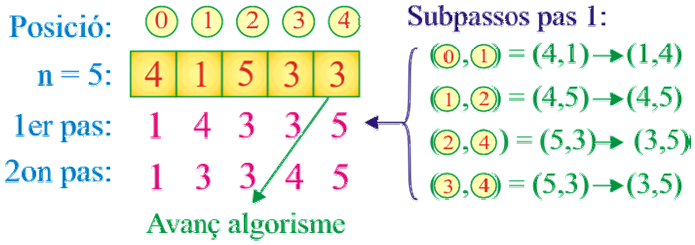
  2. En arribar a l'últim element (n) aquest ha passat a ser el més gran. Es fa, doncs, n ← n-1 com a darrer element.

  3. Es torna al pas 1 mentre en arribar a l'últim element (n) s'hagi fet algun intercanvi.
  - Per ordenació, els elements majors van quedant ordenats al final. Quan no es produeix cap canvi, vol dir que es té l'array ordenat.

6.6 Operant amb llistes (IV)

- L'algorisme queda:
  - S'ha s'observa que és un doble bucle:

```
j ← n-1
canvi ← cert
Mentre ((canvi = cert) i (j>0)) Fer //Primer bucle
    i=0
    canvi ← fals
    Mentre i<j Fer                  //Segon bucle
        Si (A[i+1] < A[i]) Llavors
            temp ← A[i]
            A[i] ← A[i+1]
            A[i+1] ← temp
            canvi ← cert
        FiSi
        i = i+1
    FiMentre
    j ← j-1
FiMentre
```

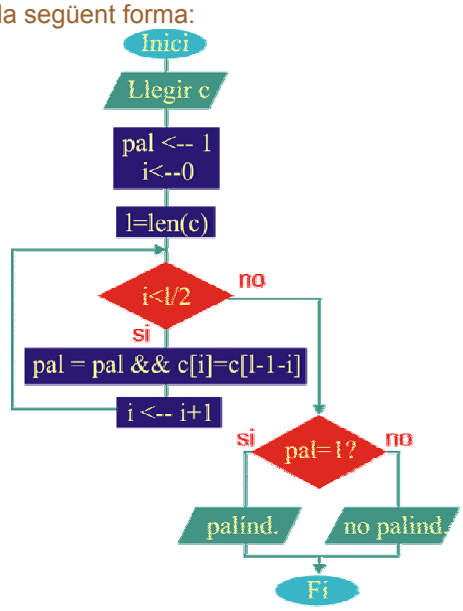


- Complexitat de l'algorisme:
  - Es fan (n-1) + (n-2) + ... 1 = n·(n+1)/2 comparacions ➔ Complexitat de l'algorisme = ϑ(n^2/2)

6.7 Exemples (I)

- Determinar si una frase és palíndrom.
  - Una cadena és palíndrom quan 'sona' igual llegida pels dos costats.
    - Exemple: ara tapat ara
  - L'algorisme, coneixent la longitud de la cadena, només ha de determinar si la meitat dreta de la frase és igual a la meitat esquerra.
  - El diagrama de flux i la codificació en C queden de la següent forma:

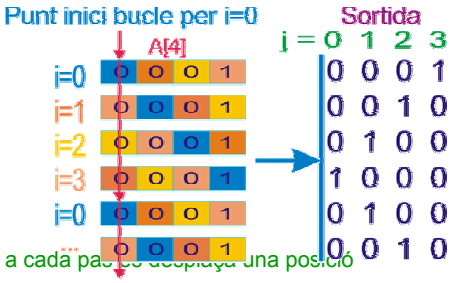
```
#include <stdio.h>
#include <string.h>
#define NCAR 80
void main()
{
    char c[NCAR];       //variable cadena
    int i, l;           //i=índex, l=longitud cadena
    int pal=1;          //pal=1 si és palíndrom
    printf("Entra un string: ");
    gets(c);
    l=strlen(c);
    for (i=0; i<l/2;i++)   pal= pal && (c[i]==c[l-1-i]);
    if (pal) printf("\n\t...es palindrom!\n\n");
    else printf("\n\t...no es palindrom!\n\n");
}
```



6.7 Exemples (II)

- Generar la seqüència 0001→0010→0100→1000→0100→0010→...
  - Aquest exercici és senzill inicialitzant un array a 0001 i aleshores provocant un desplaçament del punt d'inici de lectura de l'array, tal com indica el següent algorisme.
  - Per tal d'evitar que la seqüència vagi massa ràpid, s'ha de generar un bucle d'espera.

```
Const Enter n=10                //nº de bits
Enter A[n], i, j                //Array i índexs
Inici
    A[n-1] ← 1;                 //Array a 0...01
    i ← 0;
    Fer
        Per (j=0;j<n)
            posició ← (j+i)mod(n) //amb increment de i, a cada pas es desplaça una posició
            Escriure (A[posicio])
            EScriure('\r')      //Tornar al començament de la línia per reescriure-la
            j ← j+1
        FiPer
        i ← (i+1)mod(n)         //i marca la posició d'inici de lectura de l'array
        Esperar (0.1seg)        //temps adequat per a veure el moviment
    Mentre (i<n)                //Bucle sense sortida
Fi
```



6.7 Exemples (III)

Programa en C:

```
#include <stdio.h>
#include <time.h>
#define BITS 10             // nº bits que es mostren en pantalla
#define WAIT 10             //temps de desplaçament 1/WAIT segons
int a[BITS];                //Així s'inicialitza l'array a 0.
void sleep(clock_t);        //Declaració rutina d'espera
void main()
{
    int i=0, j;
    a[BITS-1]=1;
    do
    {
      for (j=0; j<BITS;j++) printf("%i", a[(j+i)%BITS]);   //A cada i desplaça el punt  d'inici
      printf("\r\t");
      i=(i+1)%BITS;
      sleep(WAIT);                      //bucle espera en seg/WAIT
    }while (i<BITS);
}
void sleep(clock_t fi)                  //Rutina d'espera (invers. prop. a fi)
{
    fi = clock()+CLOCKS_PER_SEC/fi;     //Constant predefinida
    while(fi>clock());
}
```

Si es substitueix la instrucció per:

```
if((inc==1 && i==BITS-1)||(inc==-1 && i==0))
        inc=(-1)*inc;
i += inc;
```

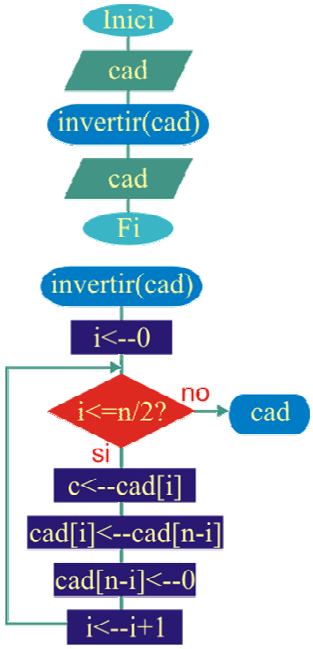
S'aconsegueix fer recircular el 1:

0001→0010→0100→1000→0100→0010→
→0001→0010→···

6.7 Exemples (IV)

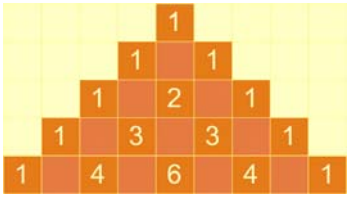
- Treballant amb cadenes. Realitzar un programa que, donada una frase, n'inverteixi l'ordre dels caràcters.

```
//Invertir l'ordre d'una frase
#include <stdio.h>
#include <string.h>              //Emprarem la funció strlen()
#define NUMCAR 80                //Núm max de caràcters
void invertir (char[], int n);   //Declaració funció inversió
void main()                      //Programa principal
{
      char cad[NUMCAR];
      printf("Entra una frase: ");
      gets(cad);
      invertir(cad, strlen(cad));
      printf("%s\n", cad);
}
void invertir (char *f, int n) //Definició funció inversió
{
      char c;
      for (int i=0; i<n/2; i++)
      {
         c=f[i];
         f[i]=f[n-i-1];
         f[n-i-1]=c;
      }
}
```

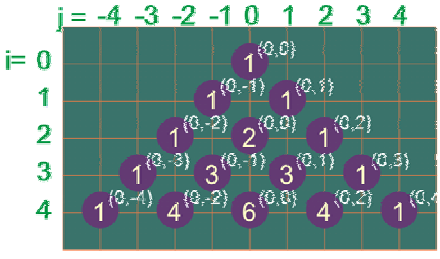


6. 7 Exemples (V)

- Generació del triangle de Pascal.
  - En el triangle de Pascal cada número és la suma dels dos que hi ha per sobre d'ell. Per exemple, el triangle de dimensió 5 (5 línies) queda:



	0	1	2	3	4
Pas 0	1	0	0	0	0
Pas 1	1	1	0	0	0
Pas 2	1	2	1	0	0
Pas 3	1	3	3	1	0
Pas 4	1	4	6	4	1

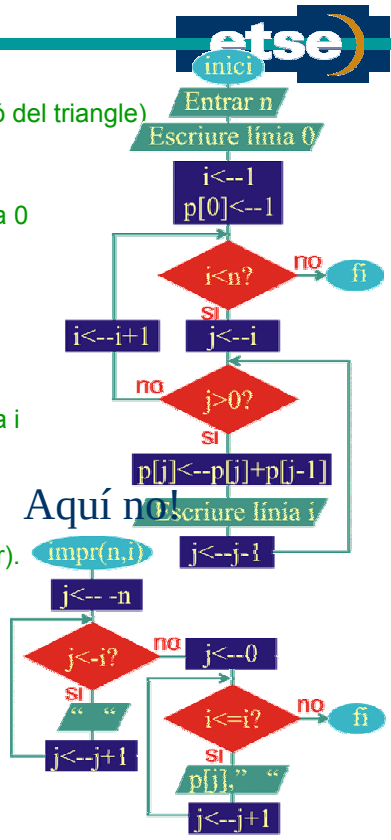


  - La generació del triangle de dimensió n és fàcil de fer amb un array de n elements. Tal com mostra la figura de la dreta: es calcula fila a fila i s'imprimeix. Segueix els passos:
    - Es comença a la fila 0 i s'acaba en la darrera (fila n-1).
    - Es comença amb un array inicialitzat en la posició 0 a 1 (és la llavor).
    - Cada fila es calcula anant des de dreta cap a esquerra, i fent a[j] ← a[j]+a[j-1]
  - L'algorisme d'impressió s'obté fàcilment a partir de la figura de la dreta:
    - L'algorisme ha d'escriure la fila i. Aleshores:
    - S'han de deixar espais des de la columna –n fins a la –i.
    - Després s'imprimeix des de la posició 0 fins a la i de l'array amb espai doble respecte al d'abans.

6. 7 Exemples (VI)

- Pseudocodi.

```
Programa principal
        //Paràmetres: p[] (array de n dimensions), n (dimensió del triangle)
        Llegir (n)              //Dimensió del triangle
        p[0] ← 1
        Imprimir(n,0)           //Crida a impressió per imprimir fila 0
        Per (i=1; i<n) Fer
            Per (j=i; j>0) Fer
                p[j] ← p[j] + p[j-1]
                j ← j-1
            FiPer
            Imprimir(n,i)   //Crida a impressió per imprimir fila i
            i ← i+1
        FiPer
Algorisme d'impressió (impressió de la fila i):
//Paràmetres passats: n (dimensió del triangle), i (fila a imprimir).
        Per (j=-n; j<-i) Fer
            Escriure (“  “);
            j←j+1
        FiPer
        Per (j=0; j<=i) Fer
            Escriure (a[j],”  “);  //Espaiat d'a[j] igual a “  “
            j←j+1
        FiPer
```



Aquí no!

6. 7 Exemples (VII)

```
/*
Triangle de Pascal. Es recomana dimensió (n) <14
*/
#include <stdio.h>
#define D 14

int p[D];               //Variable global per inicialitzar a 0
void imprimir(int, int);
void main()
{
    int i, j, n;            //n es la dimensió del triangle
    printf("Entra la dimensio (n<%i): ", D);
    scanf("%i", &n);
    printf("\n");
    p[0]=1;
    imprimir(n, 0);             //Impressió línia inicial
    for (i=1; i<n;i++)
    {
        for (j=i;j>0;j--) p[j] += p[j-1];    //Càlcul línia i
        imprimir(n, i);                 //Impressió línia i
    }
}
```

```
//Rutina d'impressió
//Imprimeix una línia del triangle
//Es passa dimensió (n), línia actual (i)
void imprimir(int n, int i)
{
    int j;
    for (j=-n; j<-i; j++) printf("   ");
    for (j=0; j<=i; j++) printf("%3i   ", p[j]);
    printf("\n");
}
```

6. 7 Exemples (VIII)

- Generar la sortida de la figura per qualsevol n (exemple per n=7):
  - És un clàssic exemple de recórrer un array.

L'algorisme imprimeix per files (índex i). Per a cada fila,

→ mentres l'índex columna (índex j) sigui menor que el i s'imprimeix j, sinó s'imprimeix i.

```
#include <stdio.h>
void main()
{
      int n;                    //dimensió de l'array
      int i, j;                 //índexs
      printf("Entra la dimensio: ");
      scanf("%i", &n);
      printf("\n\t");
      for (i=0; i<n; i++)
      {
            for (j=0;j<n;j++)
            {
                  if j<i) printf("%i", j);
                  else printf("%i", i);
            }
            printf("\n\t");
      }
}
```