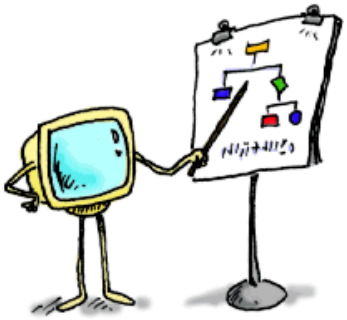
Tema 3: Iniciant la programació en C

- 3.1 El meu primer programa en C
- 3.2 Elements de C
- 3.3 Les dades en C
- 3.4 Les variables
- 3.5 Operant amb dades
- 3.6 L'entrada/sortida
- 3.7 Exemples



3.1 El meu primer programa en C

- Tots els programes en C s'ajusten a una estructura determinada.
- El següent exemple (com altres que ja s'han introduït) mostra aquesta estructura:

Directives del processador:	`#include <stdio.h>` // S'han de declarar les llibreries emprades `#define PI 3.141592` // Declaració de constants

Bloc del programador:	`/*`
	`Programa: Càlcul de l'àrea d'un cercle.`
	`Autor: Professor`
	`Descripció: Càlcul de l'àrea d'un cercle`
	`Variables: radi = Radi del cercle`
	`area = Àrea del cercle`
	`*/`

Funció principal: 	`void main()` //Programa principal
	`{`
	`float radi, area;` //Declaració de variables
	`printf("Càlcul de l'area d'un cercle.\n\n");`
	`printf("Dona'm el valor del radi → ");`
	`scanf("%f", &radi);`
	`area = PI * radi * radi;`
	`printf("\nL'area del cercle es %3.1f unitats.\n",area);`
	`}`

3.2 Elements de C

- Quan es treballa amb un llenguatge de programació s'estableix un conjunt de normes de conducta que s'han de seguir. Aquí s'introdueix, per tant, l'escenari amb el que treballa el C.
- Paraules reservades (aqui ANSI C). Són els components lèxics predefinits amb significat especial pel compilador i que no poden ser emprades més que per a la finalitat que tenen. Exemple: void, int, double, for, while, ...
- Tipus de dades. El C té tres tipus principals: nombres, caràcters i cadenes de caràcters.
- Exemple:
  - 3 → dada entera
  - 3,1416 → dada real
  - 'c'→ un caràcter
  - "Avui no plou. I demà?" → una cadena de caràcters
    - Les introduïrem més en detall en l'apartat següent.
- Les sentències. Controlen el flux d'execució d'un programa. Poden ser simples (acabades en ;) o compostes (delimitades per {}).
- L'entrada/sortida. El C realitza la lectura i sortida de dades a través de funcions específiques com ara printf(), scanf(), puts(), gets(), ...
- A recordar:
  - El programa sempre conté void main() amb el programa entre claus { }: indiquen el començament i el final del programa. Aquesta funció, única en cada programa, dirigeix l'ordre d'execució de les instruccions.
  - Cada instrucció o sentència acaba en ;
  - Els comentaris dintre la línia van després de //
  - Els comentaris entre línies comencen amb /* i acaben amb */
  - Els identificadors (noms de constants, variables o funcions) han
    - De començar per lletra o subratllat (_)
    - Han de continuar amb lletra, número o subratllat
    - Són sensitius a les majúscules.
    - No poden ser identificadors les paraules reservades
  - Convé documentar tant com es pugui un programa

3.3 Les dades en C (I)

- Classificació de les dades
  - **Segons la durabilitat dels valors:**
    - Constants → No varien el seu valor al llarg del programa. Poden ser:
    - Explícites: Exemple: b = a*2
    - Simbòliques (prèviament declarades). Exemple: p = 2*r*PI
    - Variables → Poden canviar el valor durant l'execució del programa.
  - **Segons complexitat:**
    - Simples:
      - Estàndards → Poden ser:
        - Numèriques: enters, reals.
        - Alfabètiques: caràcter, cadena.
        - Lògiques.
      - Punters → Són adreces de memòria.
      - Enumerades i subrangs.
    - Estructurades → Construïdes pel programador a partir de dades simples. Exemple: Arrays.
      - Estàtiques: Tenen la mateixa estructura al llarg del programa.
      - Dinàmiques: la seva estructura varia durant l'execució del programa.

3.3 Les dades en C (II)

- Resum del tipus de dades estàndar

Tipus de dada	Tamany (bytes)	Valor mínim	Valor màxim
signed char	1	-128	127
unsigned char	1	0	255
(signed) short	2	-32.768	32.767
unsigned short	2	0	65.535
(signed) int	2	-32.768	32.767
unsigned int	2	0	65.535
(signed) long	4	-2.147.483.648	2.147.483.647
unsigned long	4	0	4.294.967.259
float	4	3.4e-38	3.4e+38
double	8	1.7e-308	1.7e+308
long double	10	1.1e-4932	1.1e+4932

- Tots els nombres sense punt decimal es tracten com a enters, i els nombres amb punt decimal com a reals en punt flotant de doble precisió. Si es volen representar com a long, octal o hexadecimal cal posar:

	Exemple:	
0 → Octal	025	→ /* octal 25 o decimal 21 */
0x → Hexadecimal	0x25	→ /* hexadecimal 25 o decimal 37 */
L → enter llarg	250L	→ /* enter llarg 250 */

3.4 Les variables

- Les variables: declaració - definició
  - Totes les variables han de ser declarades abans de ser utilitzades. Si a més s'inicialitzen aleshores es diu que s'han definit.
  - C no té variables booleanes. Queden definides a partir d'enters:
  - si variable = 0 → fals.
  - si variable <> 0 → veritat
  - Els caràcters literals s'emmagatzemen, internament, com a int.
    - Exemples:
    - int x; //declara x com a variable entera.
    - int no = 0; //declara i defineix la variable no.
    - float total = 30.4; //Declara i defineix la variable total.
    - const float PI = 3.1416 //Declara i defineix la constant PI.
    - char car = 'M' // car té assignat el valor ASCII de M
- Cadenes de caràcters
  - Consten de 0 o més caràcters separats per dobles cometes.
  - En memòria s'emmagatzemen com a una sèrie de valors ASCII de tipus char d'1 byte, i s'acaben amb el caràcter nul.
    - char cadena[] = "Això és una cadena de caracters";
  - Els caràcters d'escapament ocupen un únic byte.
    - char c = '\n'; // nova línia → retorn de carro + nova línia

3.5 Operant amb dades (I)

- **Expressions aritmètiques.**
  - Son anàlogues a les fórmules matemàtiques: les variables i constants són numèriques i les operacions són aritmètiques.

Operador	Significat
^, ** (no en C)	Exponenciació
+, -	Suma, resta
*	Multiplicació
/	Divisió
%	Mòdul (resta)
++, --	Increment/decrement
div → div.quot , div.rem (amb llibreries)	Quocient/resta de la divisió entera

  - Fixem-nos que es distingeix entre la divisió entera i la real
  - S'estableixen les clàssiques regles de prioritat entre operadors: primer parèntesis, exponencials, multiplicació i divisió, div i mod, i, finalment, suma i resta.
    - Exemple:
      - 16 % 5 = 1
  - Compta en treballar amb variables enteres/reals!
    - Exemple: Sigui la variable real d.
      - Si es fa d=5/9 → el resultat és d=0.000000 !
      - Si es fa d=5.0/9.0 → dóna d=0.555556 !

3.5 Operant amb dades (II)

- **Operadors bit a bit .**

a b	~ (not(a))	& (and(a,b))	| (or(a,b))	^ (orex(a,b))
0 0	1	0	0	0
0 1	1	0	1	1
1 0	0	0	1	1
1 1	0	1	1	0

    - Exemple: 0x45 | 0xA3 = (01000101) | (10100011) = 11100111 = E7
  - Operadors de desplaçament: <<, >> (esquerra i dreta). Exemple: op1 = op1 << 4;
- **Operador compost**
  - L'operació aritmètica A = A + 5; és equivalent a la instrucció A += 5;
  - +=, -=, ... són operadors d'assignació compostos que permeten que el compilador generi codi de forma més eficient.
  - Els operadors d'assignació compostos es poden formar amb tots els operadors aritmètics i de manipulació de bits.
- **Operadors d'igualtat i d'assignació**

3.5 Operant amb dades (III)

- **Operadors de relació.**
  - Operadors de relació: =, <, >, <=, >=, !=
  - Exemple: Utilitzant operadors de relació .

    Amb valors numèrics:

Expressió lògica	Resultat
3 < 6	Veritat
0 > 1	Fals
9 >= 9	Veritat

    Amb caràcters → Ordre determinat per la posició dins el codi ASCII: '1'<'A', 'B'<'C'.

    Amb valors lògics: fals < veritat
    - S'ha de vigilar en comparar valors reals que no poden estar exactament emmagatzemats:
      - (1.0 / 3.0) * 3.0 pot donar 0.99... → I això podria ser considerat fals en quan el 1-lògic ha de ser 1!
- **Operadors lògics de relació.**
  - and lògica (conjunció): *(expressió1) && (expressió2)*
  - or lògica (disjunció): *(expressió1) || (expressió2)*
    - Exemples (recordar que 1 → veritat, 0 → fals):
      - float out1;
      - out1 = 0 && 1; // out1 = 0.000000
      - out1 = 1 && 3; // out1 = 1.000000
      - out1 = 0 || 1; // out1 = 1.000000
      - (5 == 5) || (6 == 7) // Vertadera (operació lògica i relacional)

3.6 L'entrada/sortida (I)

- El computador necessita de dades amb les que treballar, i una sortida a la que donar la informació processada:
  - Lectura (read) → És l'operació d'entrada de dades (pel teclat, rateta, ...)
  - Escriptura (write) → Operació de sortida de dades pel dispositiu de sortida (pantalla, impressora, ...)
- En C aquestes operacions estan implementades per les funcions printf() d'escriptura i scanf() de lectura. La funció printf().

- La funció printf()
  - S'utilitza per a escriure la informació en la sortida estàndar (normalment la pantalla).
  - És una funció d'escriptura de la biblioteca estàndard de C (stdio.h).
  - L'estructura és: printf(cadena de caràcters amb especificadors de format, variables o valors)
    - S'ha de tenir present que:
      - Els caràcters van delimitats per cometes.
      - La cadena de caràcters està delimitada per cometes dobles.
      - S'han de considerar els especificadors de format, les seqüències d'escapament i els especificadors d'ampla de camp.
  - Exemple:
    - printf("El número 11 en punt flotant seria 11.0");
      - escriuria: El número 11 en punt flotant seria 11.0
    - printf("El número %d en punt flotant és %f\n", (int)a, 11.0);
      - escriuria: El número 11 en punt flotant és 11.0000

3.6 L'entrada/sortida (II)

- En els exemples anteriors veiem especificadors de format i seqüències d'escapament.
- Especificadors de format.
  - L'especificador de format (%d, %f, ...) dóna informació sobre com es presenta la dada.
  - La sintaxi és: %<ampla_total>.<ampla_decimals>F, on F és l'especificador de format
- Seqüències d'escapament (\n, per exemple).
  - Són caràcters no imprimibles que actuen sobre el dispositiu de sortida.
  - S'indiquen amb la \ al davant.

especificadors de format

Caràcter	Argument	Significat
d,i	Enter	Enter sese signe decimal
u	Enter	Enter amb signe decimal
X	Enter	Enter sense signe hexadecimal
f	Real	Real amb signe
e	Real	Real amb signe, notació e
g	Real	Real amb signe, notació e curta
c	Caràcter	Caràcter
s	Cadena	Cadena de caràcters

seqüències d'escapament

Caràcter	Significat	Codi ASCII
\a	Caràcter d'alerta (timbre)	0x07
\b	Retrocés d'espai	0x 08
\f	Avanç de pàgina	0x 0B
\n	Nova línia	0x 0A
\r	Retorn de carro	0x 0C
\t	Tabulació (horitzontal)	0x 09
\\	Diagonal inversa	0x5B
\'	Cometa simple	0x27

- Exemples
  - printf("El valor 92 emprant d es %d.\n",92); → El valor 92 emprant d es 92.
  - printf("El valor 92 emprant X es %X.\n",92); → El valor 92 emprant X es 5C.
  - printf("El valor 92 emprant f es %3.1f.\n",92.0); → El valor 92 emprant f es 92.0.
  - printf("El valor 92 emprant e es %3.2e.\n",92.0); → El valor 92 emprant e es 9.20e+001.
  - printf("El valor 92 emprant g es %g.\n",92.0); → El valor 92 emprant e es 92.
  - printf("El valor 92 emprant c es %c.\n",92); → El valor 92 emprant c es \.

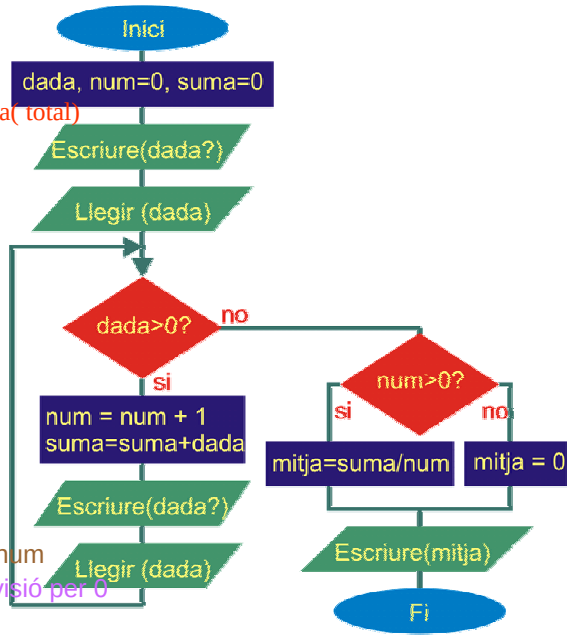
3.6 L'entrada/sortida (III)

- La funció scanf()
  - Funció de la biblioteca estàndard de C (stdio.h) que s'utilitza per a llegir de l'entrada estàndar (normalment el teclat).
  - El format és: scanf("especificadors de format", valors), on els valors són identificadors de variables que emmagatzemen les dades entrades per teclat. Això es fa emprant l'ampersand (&) al davant del nom de la variable: &valor.
  - Especificadors de format com els de la funció printf(): %c, %d, %e, %f, %u, %x.
    - Exemple: scanf("%f%d%c", &num1, &num2, &caràcter);
      - espera que siguin entrats per teclat un número real, un enter i un caràcter.
- Ús de biblioteques
  - L'ús de llibreries simplifica l'operativitat oferta per un llenguatge de programació. Algunes de les funcions matemàtiques definides en math.h i stdlib.h són:
    - int abs(int a), long int labs(long int a) → valor absolut d'a (enter)
    - double fabs(double a), long double fabsl(long double a) → valor absolut d'a (real)
    - double pow(double a1, int a2) → $a1^{a2}$
    - double exp(double a) → e^a
    - double sqrt(double a) → Arrel quadrada d'a
    - double sin/cos/tan(double a) → Funcions aritmètiques, a en radians.
    - double asin/acos/atan(double a) → Funcions aritmètiques, valors entre –1 i +1.
    - double log/log10(double a) → Logaritme neperià /decimal d'a.
    - double ceil/floor(double a) → Redondeix a l'enter major o igual/menor o igual que a.

3.7 Exemples (I)

- Calcular la mitja d'una sèrie de números enters que són entrats des de teclat.
  - La seqüència finalitza en el moment d'entrar un zero.
  - Dades:
    - Dades d'entrada: dada
    - Dades de sortida: mitja(aritmètica)
    - Variables internes: num(ero de dades), suma(total)
  - Algorisme:

```
Enter dada, num=0, suma=0
Real mitja
Inici
        Escriure ("dada?")
        Llegir (dada)
        Mentre (dada>0) Fer
                num = num+1
                suma=suma+dada
                Escriure ("dada?")
                Llegir (dada)
        FiMentre
        Si (num>0) aleshores mitja=suma/num
        SiNo mitja=0            //evita la divisió per 0
        Escriure ("Mitja = ", mitja)
Fi
```



3.7 Exemples (II)

- Calcular la mitja d'una sèrie de números enters que són entrats des de teclat.
  - La seqüència finalitza en el moment d'entrar un zero.
  - El diagrama de flux és el que s'ha realitzat en el capítol anterior.
  - El programa en C és el següent (es pot observar clarament el format: directives, explicació de l'algorisme, declaració de variables i el programa.

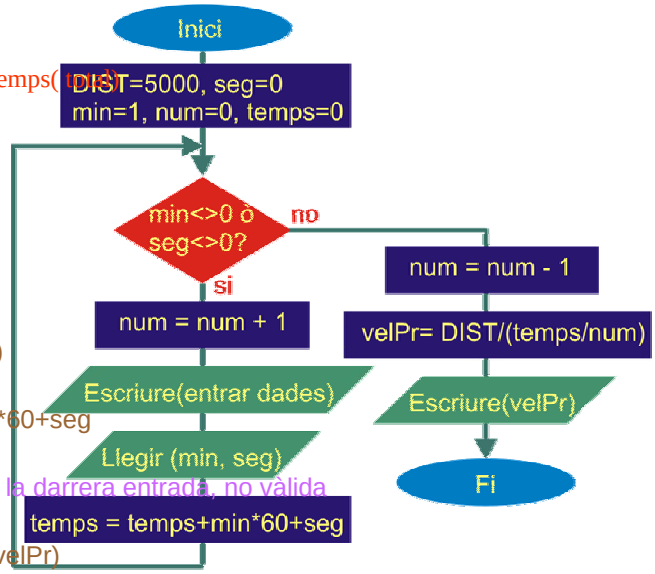
```
#include <stdio.h>
/* Calcul de la mitja aritmetica d'un conjunt de numeros enters positius entrats des de teclat. S'acaba en
    entrar un 0 o un nombre negatiu.*/
void main()
{
    int dada, num=0, suma=0;
    float mitja;
    printf(" Entra un nou numero: ");          //Nou numero
    scanf("%i", &dada);
    while (dada>0)
    {
        num++;                                 //Index numeros entrats
        suma += dada;                          //Suma actual
        printf(" Entra un nou numero: ");
        scanf("%i", &dada);
    }
     if (num>0) mitja = suma*1.0/num;          //Mitja aritmètica: fixar-se en ...*1.0!!
    else mitja=0;                              //Cas de no entrar cap numero
    printf(" La mijta aritmètica dels %u nombres entrats es %3.2f\n", num, mitja);
}
```

3.7 Exemples (III)

Calcular la velocitat promig d'un conjunt de corredors d'atletisme de 5000m.

- Les dades s'entraran en min i seg per corredor. La sortida serà la velocitat promig en m/s.
- Quan no quedi cap més corredor s'entrarà el temps 0min 0seg.
- Dades:
  - Dades d'entrada: DIST, min, seg
  - Dades de sortida: velPr(omig)
  - Variables internes: num(ero d'atletes), temps(total)
- Algorisme:

```
Constant DIST=5000
Enter seg=0, min=1, num=0, temps=0
Real velPr
Inici
    Mentre (min>0 ò seg>0) Fer
                num = num+1
                Escriure ("min, seg")
                Llegir (min, seg)
                temps = temps+min*60+seg
        FiMentre
        num=num-1              //Es treu la darrera entrada, no vàlida
        velPr = DIST/(temps/num)
        Escriure ("Velocitat promig = ", velPr)
Fi
```



- Falla en algun cas aquest algorisme??

3.7 Exemples (IV)

- Programa en C:

```
#include <stdio.h>
#define DIST 5000
/* Calcul vel. promig corredors 5000m*/
void main()
{
    int  seg=0, min=1, num=0, temps=0;
    float velPr;
    while (min>0 || seg>0)
    {
        num++;
        printf("Entra el temps del corredor %i (min seg): ", num);
        scanf("%i%i", &min, &seg);
        temps += min*60+seg;
    }
    num--;                                  //descomptar el corredor amb temps 0 0
    velPr  = DIST*1.0 / (temps / num);      //velocitat promig
    printf("Velocitat promig dels %i corredors: %2.1f m/s\n", num, velPr);
}
```