APROFUNDINT EN L'ENTRADA/SORTIDA

1. Eines que disposa el llenguatge C per a tractar l'E/S

1.1 E/S per pantalla (o consola)

Recordem que en C no existeix cap instrucció per a realitzar l'entrada i sortida de dades (E/S). Aquesta s'ha de fer utilitzant els programes que proporciona la biblioteca stdio.

Els programes que realitzen l'E/S amb format, és a dir, poden llegir i escriure la informació de diverses formes controlades, són:

- **printf():** Escriu dades en la pantalla amb el format especificat.
- **scanf()**: Llegeix dades per teclat amb el format especificat.

Sortida de dades: El programa printf()

La manera d'utilitzar aquesta programa és la següent:

```
printf("cadena_control",llista_arguments);
```

Al programa printf() a `cadena_control`, a més a més del text que es vulgui mostrar per pantalla, es poden posar tant caràcters d'**escapament** com **especificadors de format** (coneguts també com caràcters de transmissió). Recordem que a la `llista_arguments` han de figurar els noms de les variables que contenen les dades que volem mostrar per pantalla.

Els caràcters d'**escapament** tenen un significat especial:

\v Tabulador vertical.
\0 Caràcter nul.
\n Nova línia.
\t Tabulador horitzontal.
\b Retrocedir un caràcter.
\r Retorn de carro.
\f Salt de pàgina.
\' Imprimeix la cometa simple.
\" Imprimeix la cometa doble.
**** Imprimeix la barra invertida '\'.
\xdd dd es el codi ASCII, en hexadecimal, del caràcter que es desitja imprimir.

Quan en un text a imprimir es desitja intercalar el valor d'una variable, en la posició on hauria d'anar el valor es col·loca el caràcter % seguit d'alguns caràcters. Depenent dels caràcters que s'introdueixin, s'imprimirà un valor d'un tipus de dades concret, amb un format de presentació determinat. El caràcter % i els caràcters que el segueixen són el que s'anomenen **especificadors de format (**o caràcters de **transmissió)**.
Al final de la cadena de control, després de las cometes de tancament, es col·loca una coma i a continuació va la llista d'arguments, aquí indicarem el nom de la variable que es desitja imprimir.
Els **especificadors de format** tenen la forma:

%[flags][ample de camp][.precisió][F/N/h/l/L] **tipus**

Les dades entre **[**...**]** són opcionals.

[flags]: Son caràcters que introdueixen modificacions en el mode en que es presenta el valor. Alguns del seus valors i significats són:

- caràcter '–': el valor queda justificat a la esquerra.
- caràcter +: el valor s'escriu amb signe, sigui positiu o negatiu.
- caràcter en blanc: Si el valor numèric és positiu, deixa un espai en blanc. Si és negatiu imprimeix el signe.

[ample de camp][.precisió]
Aquest dos indicadors opcionals han d'anar abans dels indicadors de tipus de dades. Amb l'amplada de camp, l'especificador de format indica al programa printf la longitud mínima que ha d'ocupar la impressió del valor que s'ha de mostrar.
El paràmetre de precisió s'utilitza per valors en coma flotant. Indica el número de decimals que s'han de mostrar.

[F/N/h/l/L] Aquestes cinc lletres són modificadors de tipus i precedeixen a les lletres que indiquen el tipus de dades que s'ha de mostrar per pantalla.
La lletra **h é**s el modificador ***short*** per valors enters.
La lletra **l** té 2 significats: és el modificador ***long*** per valors enters. I, precedint a la lletra f indica que allí ha d'anar un valor de tipus ***double***.
La lletra **L** precedint a la lletra f indica que allí ha d'anar un valor de tipus long double.

Tipus És l'únic argument necessari. Consisteix en una lletra que indica el tipus de dades a que correspon el valor que es desitja imprimir.

%d	Enter amb signe, en base decimal.
%i	Enter amb signe, en base decimal.
%o	Enter sense signe, en base octal.
%u	Enter sense signe, en base decimal.
%x	Enter sense signe, en base hexadecimal, amb lletres minúscules.
%X	Enter sense signe, en base hexadecimal, amb lletres majúscules.
%f	Número real amb signe.
%e	Número real amb signe en format científic, amb l'exponent 'e' en minúscula.
%E	Número real amb signe en format científic, amb l'exponent 'e' en majúscula.
%g	Número real amb signe, a elegir entre format e ó f segons la mida del valor.
%G	Número real amb signe, a elegir entre format E ó f segons la mida del valor.
%c	Un caràcter. El caràcter llur ASCII correspongui amb el valor a imprimir.
%s	Cadena de caràcters.
%p	Adreça de memòria.
%	Si el caràcter % no va seguit de res, llavors s'imprimeix el caràcter sense més.

Ex: `int i;  printf("El valor de la dada es: %d",i);`

Entrada de dades: El programa scanf()

La manera d'utilitzar aquesta programa és la següent:

```
scanf("cadena_control",llista_de_variables);
```

Permet la entrada de dades des del teclat. L'execució d'aquest programa queda suspesa en espera de que l'usuari introdueixi un valor pel teclat i premi la tecla de validació (intro).

Amb la `cadena_control` s'indica el tipus de dada del valor que s'espera rebre per teclat. El tipus s'indica amb un especificador de format, a continuació teniu alguns exemples:

%d	Enter amb signe, en base decimal.
%i	Enter amb signe, en base decimal.
%o	Enter sense signe, en base octal.
%u	Enter sense signe, en base decimal.
%x	Enter sense signe, en base hexadecimal, usant lletres minúscules.
%f	Número real amb signe.
%e	Número real amb signe en format científic, amb l'exponent 'e' en minúscula.
%c	Un caràcter. El caràcter llur ASCII correspon amb el valor a llegir.
%s	Cadena de caràcters.
%p	Direcció de memòria.

A més de la cadena de control, el programa scanf() requereix d'una `llista_de_variables` on cal indicar el nom de la variable a on s'ha d'emmagatzemar el valor introduït pel teclat (com passa amb el programa printf) precedit de l'operador **&** .

Ex: `int i;` `scanf("%d",&i);`

L'oblit d'aquest operador és freqüent en programadors novells!.

El programa scanf() pot llegir del teclat tantes entrades com se li indiquin. De totes maneres, es recomana usar scanf() per a cada entrada diferent que es requereixi.

Ex: `scanf("%d %d",&i,&j);` es equivalent a `scanf("%d",&i); scanf("%d",&j);`

1.2 Introducció als fitxers

A l'assignatura treballarem amb dos tipus de fitxers bàsics:

(i) els **fitxers de text** on la seva característica principal és que guarden tota la informació com a caràcters i per tant contenen dades llegibles per una persona i es poden generar o modificar des dels programes o usant editors de textos.

(ii) els **fitxers binaris** on la seva característica és que la informació es guarda de la mateixa forma que en la memòria del computador, és a dir, un caràcter es guarda segons el seu codi ASCII (codi numèric), però un enter es guarda segons la seva codificació binària. Això implica que per a generar i/o modificar el fitxer s'hagi de fer a partir de programes que interpretin el seu format.

Si guardem un valor de tipus enter en un fitxer de text, per exemple el valor 12, realment el que estem emmagatzemant en el fitxer de text són els dígits que formen el valor, en aquest cas el dígit 1 i el dígit 2, és a dir, dos dades de tipus caràcter. A l'hora de llegir la dada, podrem fer-ho amb qualsevol variable de tipus enter amb capacitat suficient per a emmagatzemar el valor (int, unsigned int, char, unsigned char, ...). Observeu que, codificat com a text 12 ocupa dos bytes, però que si s'emmagatzema en una variable de tipus int ocupa 4.

Fixeu-vos també que ocuparem diferent espai si guardem el valor enter 12 (com a short int: 2 bytes) de si guardem el valor enter 139.247 (com a int: 4 bytes).

Un problema dels fitxers de text és la necessitat d'usar marques de separació entre els seus diferents elements. Si, per exemple al valor 12 li segueix el valor 100, no podem limitar-nos a disposar un a continuació de l'altre, ja que el fitxer tindria la següent seqüència de caràcters 12100. Què estem representant amb aquesta informació exactament? Un 12 seguit d'un 100 o un 1 seguit d'un 2100? I per què no un 1210 seguit d'un 0 o, senzillament, el valor 12100?

Les marques de separació són caràcters que decideix el programador, però normalment s'utilitza l'espai en blanc i en menys mesura, tabuladors o salts de línia.

Es pot representar:

12 100
12 \t 100
12 \n 100

Els caràcters separadors també ocupen espai en el fitxer de text (són també caràcters).

Els fitxers binaris guarden la informació tal i com està representat en memòria, per tant es guardarà diferent si escric en el fitxer el caràcter de codi 12, l'enter de valor 12 o el real de valor també 12. Els bytes que s'ocuparan en cada cas també seran diferents i dependran del tipus de variable. Si guardem el 12 com un char, guardarem un sol byte format per aquests 8 bits: **00001100** Però si optem per emmagatzemar-lo com un int, seran 4 bytes escrits: **00000000 00000000 00000000 00001100**

Un mateix patró de 8 bits, com 11111111 té dos possibles interpretacions: el valor 255 si entenem que es una dada de tipus unsigned char o el valor -1 si considerem que es codifica un char.

Com es pot veure, la seqüència de bits que escrivim en el fitxer és exactament la mateixa que hi ha emmagatzemada a memòria, utilitzant la mateixa codificació binaria, per això s'anomenen fitxers binaris.

Escriure dues o més dades d'un mateix tipus en un fitxer binari no requereix la inserció de marques separadores: cada cert número de bytes comença una nova dada (cada 4 bytes per exemple comença un nou int) així és fàcil decidir on comença i acaba cada dada.

La lectura d'un fitxer binari requereix del coneixement exacte del tipus de dades de cada un dels valor emmagatzemats en ell, doncs en cas contrari la seqüència de bits no tindria el significat amb el que s'ha guardat.

Els fitxers binaris no només poden emmagatzemar escalars, també registres i vectors, que aprendrem a utilitzar en els següents temes.

Per regla general, els fitxers binaris són més compactes que els fitxers de text, cada valor ocupa el mateix que ocuparia en memòria, el valor enter 12 ocupa el mateix que el valor enter 139.247 (4 bytes en tots dos casos). La lectura (i escriptura) de les dades en fitxers binaris també es més ràpida, ens estalviem el procés de conversió del format text a la representació d'informació en memòria i viceversa, però per altra banda són menys portables que els fitxers de text i per a manipular-los sempre depenem d'un programa que els interpreti.

1.3 Fitxers de text

Els fitxers de text es manipulen seguint aquest procediment:

1. Es defineix una variable de tipus fitxer.
2. S'obre el fitxer en mode lectura, escriptura, afegir o qualsevol altre mode vàlid.
3. Es treballa amb el fitxer llegint o escrivint dades, segons el mode d'apertura escollit.
4. Es tanca el fitxer.

Recordem els programes amb els que hem treballat en pseudocodi són:

- **obrir**, **llegirf**, **escriuref** i **tancar**.

Afegirem el nou programa **farxiu** que es descriu més endavant.

En C disposem dels programes equivalents: **fopen**, **fscanf**, **fprintf, fclose** i l'equivalent de farxiu és **feof**. A continuació trobareu com s'utilitzen aquestes programes.

- **fopen:** obre un fitxer. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `nom_var_fitxer=fopen("ruta\nom_arxiu","modus");`

  Entre parèntesis primer cal indicar el nom del fitxer amb la seva localització (ruta) i a continuació el mode d'apertura. Els modes d'apertura poden ser:

  r: Obre un arxiu de text per lectura
  w: Crea un arxiu de text per escriptura
  a: Obre un arxiu de text per afegir (append)
  r+: Obre un arxiu de text per lectura/escriptura
  w+: Crea un arxiu de text per lectura/escriptura
  a+: Afegeix o crea un arxiu de text per lectura/escriptura

  Ex: `FILE *f;` `f=fopen("C:\carpeta\nom_arxiu","r");`

- **fclose:** tanca el fitxer. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `fclose(nom_var_fitxer);`
  Entre parèntesis cal indicar la variable (tipus fitxer) que ha estat prèviament modificada pel programa `fopen`.

Cada operació d'obrir un fitxer amb fopen ha d'anar acompanyada del programa fclose una vegada s'ha acabat de treballar amb el fitxer.

- **fscanf:** llegeix d'un fitxer. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `fscanf(nom_var_fitxer,"format",llista_de_variables);`
  Entre parèntesis cal indicar la variable (tipus fitxer) que ha estat prèviament modificada pel programa fopen. A continuació el format i per últim a on es guardaran els valors llegits (de la mateixa manera que amb el programa scanf). El programa a més a més proporciona el número d'elements llegits.

  Ex: `FILE *f; int i, n;` `n=fscanf(f,"%d",&i);`
  n contindrà els nombre d'elements llegits.

- **fprintf:** escriu en un fitxer. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `fprintf(nom_var_fitxer, "cadena_control",llista_arguments);`
  Entre parèntesis cal indicar la variable (tipus fitxer) que ha estat prèviament modificada pel programa fopen. A continuació el format i per últim els valors que volem escriure. El programa a més a més proporciona el número de caràcters que s'han escrit.

  Ex: `FILE *f; int i, n;` `n=fprintf(f,"%d",i);`
  n contindrà els nombre de caràcters escrits.

- **feof:** El nom del programa és l'abreviatura de *end of file*. El programa ens proporciona un valor que ens indica si estem o no al final del fitxer, només tindrà sentit fer-lo servir després de realitzar una lectura de dades. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;` `feof(nom_var_fitxer)`
  Entre parèntesis cal indicar la variable tipus fitxer (que ha estat prèviament modificada pel programa fopen).
  Ex: `FILE *f;` `int n;` `n=feof(f);`
  n contindrà un 1 si estem al final del fitxer i 0 en cas contrari.

Aquest programa també es pot utilitzar en arxius binaris.

La operació equivalent en pseudocodi és `farxiu` i la manera d'utilitzar-la és:
Ex: `var f: fitxer;  b: booleà;`
`b:= farxiu(f);`
Si estem al final del fitxer b contindrà cert i fals en cas contrari.

1.4 Fitxers binaris

La gestió de fitxers binaris obliga a treballar amb el mateix procediment bàsic:

1. definir la variable de tipus fitxer.
2. obrir el fitxer en el mode adequat.
3. llegir i/o escriure informació.
4. tancar el fitxer.

Per obrir un fitxer binari s'utilitza el mateix programa que l'utilitzat pels fitxers de text. El que canvia és el *mode*, en aquest cas està format pel mateix identificador seguit de la lletra **b**. Els modes d'apertura bàsics per a fitxers binaris són:

- "rb" (lectura): El primer byte llegit es el primer del fitxer.
- "wb" (escriptura): Trunca el fitxer a longitud 0. Si el fitxer no existeix, es crea.
- "ab" (afegir): És un mode d'escriptura que preserva el contingut original del fitxer. Les dades escrites s'afegeixen al final del fitxer.

El programa de tancament del fitxer és el mateix que pels fitxers de text.

En pseudocodi per llegir i escriure als fitxers binaris utilitzarem els mateixos programes que en fitxers de text, l'únic que a l'hora d'obrir l'arxiu caldrà especificar el mode d'apertura binari.

Per llegir i/o escriure a fitxers binaris en C cal utilitzar els següents programes:

- **fread:** llegeix d'un fitxer binari. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `fread(variable,mida_dada,nombre_dades,nom_var_fitxer);`
  Entre parèntesis cal indicar la variable on es guardarà la dada llegida del fitxer precedida del caràcter **&**, a continuació a `mida_dada` el número de bytes que ocupa una dada, a `nombre_dades` el número de dades a llegir i finalment el fitxer. El programa a més a més proporciona el número de dades que ha aconseguit llegir (si aquest valor es menor que `nombre_dades`, es perquè hem arribat al final del fitxer i no s'ha pogut efectuar la lectura completa).

  Ex:
  ```
  int a, n; FILE * fit;
  n=fread(&a,sizeof(int),1,fit);
  ```

  Es C es disposa del programa **sizeof** que proporciona la mida del tipus de dada que s'indica entre parèntesis.

- **fwrite:** escriu a un fitxer binari. La manera d'utilitzar aquesta programa és la següent:
  Ex: `FILE *nom_var_fitxer;`
  `fwrite(variable,mida_dada,nombre_dades,nom_var_fitxer);`
  Entre parèntesis cal indicar la variable que conté el valor que s'escriurà al fitxer precedida del caràcter **&**, a continuació a `mida_dada` el número de bytes que ocupa una dada, a `nombre_dades` el número de dades a escriure i finalment el fitxer. El programa a més a més proporciona el número de dades que s'han aconseguit escriure.

  Ex:
  ```
  int a, n; FILE * fit;
  n=fwrite(&a,sizeof(int),1,fit);
  ```