5. Funcions d'entrada/sortida: stdin, stdout

Introducció

Ja hem comentat que els programes són manipuladors d'informació. Necessiten obtenir una informació per processar-la i poder donar sortida als resultats. Si no hi ha entrada de dades, els programes es limiten a fer sempre el mateix amb les mateixes dades. Si no hi ha sortida de dades, els resultats del programa no es poden usar per a res.

El concepte d'entrada de dades és força ampli. Es pot referir a les tecles que es premen en un teclat, la posició del ratolí o de la pantalla tàctil, un fitxer en disc, una adreça d'internet, un lector de codis de barres, una webcam o un micròfon. De la mateixa manera, quan parlem de sortida, ens podem referir a un ampli ventall d'opcions, des de la pantalla de l'ordinador fins a un arxiu, una impressora o el sistema de control d'un robot.

Entre tots aquests dispositius, n'hi ha dos que tenen una funció especialment rellevant: el **teclat**, per a l'entrada de dades, i la **pantalla**, per a la sortida de dades. De fet, si no es canvia explícitament, l'entrada estàndard (*standard input* o *stdin*) és el teclat, i la sortida estàndard (*standard output* o *stdout*) és la pantalla.

En aquesta guia, veurem com ens podem comunicar amb aquests dispositius per llegir la informació des del teclat i mostrar-la en la pantalla.

Compte!

En aquests apartats, esmentarem els conceptes de **cadena de caràcters (***string***)** i **punter** o **adreça de memòria**, que es desenvoluparan més endavant en profunditat. En aquest punt, no cal que els entengueu, però els afegim perquè, un cop els conegueu, pugueu utilitzar aquests materials com a referència.

1. Sortida de dades per pantalla: *standard output*, *stdout*

La manera més simple de mostrar dades per pantalla és escriure-les com si fossin text (evitant imatges, etc.). Per fer-ho, es disposa d'una sèrie de funcions que enviaran la informació que volem mostrar a la pantalla. Una **funció** és una paraula que ha estat definida abans i que el compilador entén, la qual cosa ens permet fer accions; en aquest cas, mostrar text per pantalla. Es pot donar el cas que una funció rebi variables com a paràmetres a l'hora de cridar-les. Hi ha moltes funcions que anirem descobrint al llarg del curs.

Hi ha diferents funcions que mostren text per pantalla. Són les següents:

Funció	Descripció
writeChar(ch);	Escriu en pantalla el caràcter guardat de la variable *ch*, que ha de ser de tipus **char**.

writeBoolean(bo);	Escriu en pantalla el booleà guardat de la variable *bo*, que ha de ser de tipus **boolean**.
writeInteger(in);	Escriu en pantalla l'enter guardat de la variable *in*, que ha de ser de tipus **integer**.
writeReal(re);	Escriu en pantalla el real guardat de la variable *re*, que ha de ser de tipus **real**.
writeEnum(en);	Escriu en pantalla el valor enumeratiu guardat de la variable *en*, que ha de ser de tipus **enum**.
writeString("Character string");	Escriu en pantalla una cadena de caràcters. La cadena va escrita entre cometes.

A continuació, teniu un breu pseudocodi que mostra com s'usen aquestes funcions.

```
type
   exampleEnum = {VALUE1, VALUE2, VALUE3}
end type

algorithm sumStdout

   var
      c: char ;
      b: boolean ;
      i1, i2: integer ;
      f: real ;
      en: exampleEnum;
   end var

   c := 'a';
   b := true;
   i1 := 2;
   i2 := 2;
   f := 6.68;
   en := VALUE2;

   writeString("The character stored in c is : ");
   writeChar(c);

   writeString("The boolean stored in b is : ");
```

```
   writeBoolean(b);

   writeString("The sum of the integers i1 and i2 is:");
   writeInteger(i1 + i2);

   writeString("The real stored in f is : ");
   writeReal(f);

   writeString("The value stored in :");
   writeEnum(en);
end algorithm
```

1.1. Sortida de dades per pantalla en C

Per mostrar dades per pantalla en C, utilitzarem la funció `printf`. Aquesta funció executa totes les funcions de sortida que s'han presentat en l'apartat anterior. En la taula següent, podeu observar com *ch* és una variable `char`, *bo* és una variable `bool`, *int* és una variable `integer` i *re* és una variable real/`float`:

Pseudocodi	C
writeChar(ch);	`printf("%c", ch);`
writeInteger(in);	`printf("%d", in);`
writeBoolean(bo);	`printf("%d", bo);`
writeReal(re);	`printf("%f", re);`
writeEnum(en);	`printf("%u", en);`
writeString("String of characters");	`printf("String of characters");`

Si us hi fixeu, la funció `printf` es crida utilitzant dos paràmetres:

- El primer, entre cometes, és un descriptor de format:
  - Per escriure un caràcter cal escriure «%c».
  - Per escriure un `int`, «%d» (que es refereix a decimal).
  - Per a un `bool`, també utilitzarem «%d» (imprimirà 1 o 0 per pantalla).
  - Per un `float`, «%f».
  - Per escriure un `enum`, escriurem l'índex sencer del qual correspon. Utilitzarem «%u», que es refereix a *unsigned int* (enter sense signe).
- El segon és la variable que s'imprimirà. De fet, podria ser més d'una variable, depenent del descriptor de format. Ara com ara, ho simplificarem a una sola variable.
- Si voleu escriure una cadena de caràcters, simplement l'heu d'escriure entre cometes en comptes del descriptor de format, i no cal posar-hi una variable.

A continuació, s'adjunta una taula que identifica l'especificador de format que correspon a cada tipus de dada:

Tipus de dada	Especificador de tipus	Exemple de dades
Caràcter	%c	'a'
Booleà	%d	0 (equivalent de `false`), 1 (equivalent de `true`)
Enumeratiu	%u	{VALUE1, VALUE2, ... VALUEN}
Enter	%d	34 o -32
Real	%f	3.1415
String	%s	"Hello World"
Punter	%p	0x000000AA (els **punters** els tractarem en la unitat 19. Punters en C).

De vegades, ens interessarà especificar exactament el nombre de xifres que posem en un nombre, el nombre de decimals, etc. Això es pot fer amb els modificadors de format. Aquests **modificadors** són valors extres que s'afegeixen als especificadors (%d, %b, %f, etc.). Tot seguit teniu un resum de com s'usen aquests modificadors:

%[modificadors][ample][.precisió]especificador

A continuació, en podeu veure uns quants exemples. Trobareu més informació a l'**enllaç següent (http://www.tutorialspoint.com/c_standard_library/c_function_printf.htm)**.

Format	Valor de la variable	Sortida	Descripció
"%d"	3	3	Aquest és el resultat d'imprimir un nombre enter sense modificador.
"%03d"	3	003	El modificador 03 fa que s'afegeixin zeros a l'esquerra del nombre fins a arribar a tres xifres.
"%f"	3.14	3.14000	Aquest és el resultat d'imprimir un nombre real sense modificador.
"%2.1f"	3.14	3.1	El modificador 2.1 fa que la longitud mínima del nombre que s'ha d'imprimir sigui de dos caràcters i que només hi hagi un decimal.

Tenint en compte el que hem vist fins ara, ja podem traduir a C el pseudocodi que hem escrit en l'apartat anterior:

Exemple 01

Pseudocodi	C

algorithm sumStdout
 	
var
ch: **char** ;
bo: **boolean ;**
in1, in2: **integer ;**
re: **real ;**
end var
 	
ch := 'a';
bo := **true**;
in1 := 2;
in2 := 3;
re := 2.0;
	`#include <stdbool.h>`
	`#include <stdio.h>`
	
	`int main() {`
	`  char ch;`
	`  bool bo;`
	`  int in1, in2;`
	`  float re;`
	
	`  ch = 'a';`
	`  bo = true;`
	`  in1 = 2;`
	`  in2 = 3;`
	`  re = 2.0;`
	
	`  printf("The character stored in ch is:");`
	`  printf("%c \n", ch);`

writeString("The character stored in ch is : ");
writeChar(c);
 	
writeString("The boolean stored in bo is : ");
writeBoolean(b);
 	
writeString("The sum of the integers in1 and in2 is:");
writeInteger(in1 + in2);
 	
writeString("The real stored in re is : "):
writeReal(f);
 	
end algorithm
	`printf("The character stored in bo is:");`
	`printf("%d \n", bo);`
	
	`printf("The sum of in1 and in 2 is:");`
	`printf("%d \n", in1 + in2);`
	
	`printf("The real stored in re is:");`
	`printf("%f \n", re);`
	
	`return 0;`
	`}`

Recordeu

Per utilitzar la funció `printf`, heu d'afegir una llibreria al programa. Les **llibreries** són conjunts de funcionalitats que estenen les funcions bàsiques del llenguatge C, de manera que faciliten certes tasques. Per utilitzar `printf`, hem d'afegir en el preprocessador la línia següent:

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

Encara podem simplificar aquest programa una mica més. En usar `printf`, l'especificador de format pot incloure alhora cadenes de caràcters i diverses variables.

Exemple 02

Escriure més d'una variable en una sola crida a `printf`:

Pseudocodi	C
algorithm sumStdout	`#include <stdio.h>` `#include <stdbool.h>`

```
var
   ch: char;
   bo: boolean;
   in1, in2: integer;
   re: real;
end var

ch := 'a';
bo := true;
in1 := 2;
in2 := 3;
re := 2.0;

writeString("The character stored in ch is : ");
writeChar(ch);

writeString("The boolean stored in bo is : ");
writeBoolean(bo);

writeString("The sum of the integers ");
writeInteger(in1);
writeString(" and ");
writeInteger(in2);
writeString(" is: ");
writeInteger(in1 + in2);

writeString("The real stored in re is : ");
writeReal(re);

end algorithm
```

```c
int main() {
    char ch;
    bool bo;
    int in1, in2;
    float re;

    ch = 'a';
    bo = true;
    in1 = 2;
    in2 = 3;
    re =  2.0;

    printf("The character stored
in ch is: %c \n", ch);

    printf("The character stored
in bo is: %d \n", bo);

    printf("The sum of %d and %d
is: %d \n", in1, in2, in1 + in2);

    printf("The real stored in re
is: %f \n", re);

    return 0;
}
```

Analitzem la línia següent del programa C:

```
writeString("The sum of the integers ");
writeInteger(in1);
writeString(" and ");
writeInteger(in2);
```

```c
printf("The sum of %d and %d is:
%d\n", in1, in2, in1 + in2);
```

writeString(" is: "); **writeInteger**(in1 + in2);	
	La crida a `printf` té diversos paràmetres: • El format: que descriu la informació que s'imprimirà. En aquest cas, és una mescla de diverses cadenes de caràcters (`"The sum of"`, `"and"` i `"is:"`) i variables (tres enters, `"%d"`). • Tres variables: com que el format inclou tres variables enteres, cal afegir les tres variables que s'escriuran en pantalla. En aquest cas són les següents: ◦ `in1` ◦ `in2` ◦ `in1 + in2`

2. Entrada de dades per teclat: *standard input, stdin*

Quan estem generant pseudocodi en dissenyar un programa, hi ha moltes funcions que podem usar per llegir la informació des del canal estàndard, és a dir, des del teclat. En aquest punt, us presentem les funcions que llegeixen els diferents tipus de dades des del teclat. Com en el cas de *stdout*, aquestes funcions depenen de la mena de dada que volem llegir:

Funció	Descripció
ch := **readChar**();	Llegeix un caràcter per teclat i el guarda en la variable char ch.
bo := **readBoolean**();	Llegeix un booleà per teclat i el guarda en la variable bool bo.
in := **readInteger**();	Llegeix un enter per teclat i el guarda en la variable integer in.
re := **readReal**();	Llegeix un real per teclat i el guarda en la variable real re.
en := **readEnum**();	Llegeix un enumeratiu per teclat i el guarda en la variable enum en.

Fixeu-vos en què *ch* és una variable **char**, *bo* és una variable **boolean**, *int* és una variable **integer** i

re és una variable **real**. Aquestes variables han d'haver estat definides abans de cridar les funcions de lectura pel canal estàndard.

2.1. Entrada de dades per teclat en C

En C, el mètode genèric per a l'entrada de dades per teclat és `scanf`. En la taula següent, resumim la manera d'utilitzar-lo:

Pseudocodi	C
ch := **readChar**();	`scanf("%c", &ch);`

bo := **readBoolean**();	Per llegir un booleà des del teclat, ho farem de la manera següent:
	`bool bo;`
	`int intToBool;`
	`scanf("%d", &intToBool);`
	`bo = (bool)intToBool;`
	El llenguatge C no disposa de cap especificador de tipus que treballi només amb 1 bit, de manera que utilitzarem una variable auxiliar que ajudi a fer una conversió intermèdia a `int`, per tal de transformar posteriorment el valor a `bool`.
	En cas de no llegir els booleans d'aquesta manera, poden donar-se dues situacions diferents, segons el compilador de C que utilitzem:
	• Que la variable de tipus `bool` no contingui el valor esperat, la qual cosa ocasionarà que el nostre programa es comporti de manera incorrecta.
	• Que es generi un avís del tipus següent: `warning: format '%d' expects`

	`argument of type 'int *'; but argument 2 has type '_Bool *' [-Wformat=]`. Aquest avís significa que estem utilitzant un especificador de tipus (%d) diferent del que li correspondria al tipus `bool`, que treballa únicament amb 1 bit.
in := **readInteger**();	`scanf("%d", &in);`
re := **readReal**();	`scanf("%f", &re);`
en := **readEnum**();	`scanf("%u", &en);` En aquest cas, es llegeix un enter que correspon a l'índex de l'`enum`. És a dir, no llegim l'`enum` directament, sinó mitjançant el seu índex.
st := **readString**();	`scanf("%s", st);`

Fixeu-vos en què:

- *ch* és una variable `char`.
- *bo* és una variable `bool`.
- *in* és una variable `int`.
- *re* és una variable `float`.

Aquestes variables han d'haver estat definides abans de cridar les funcions de lectura pel canal estàndard.

De manera semblant a com utilitzàvem `printf`, la funció `scanf` espera els paràmetres següents:

- L'especificació del format de la informació que es llegirà pel teclat. Aquest format s'especifica de la mateixa manera que per a la funció `printf`, com ja hem explicat en l'apartat anterior.
- Les variables en què es guardaran els valors llegits. En el cas de la funció `scanf`, aquestes variables (excepte en el cas de *string*) han d'anar precedides del símbol &. Treballarem l'operador & en les unitats següents, però us avancem que significa 'adreça de' i que es refereix a l'adreça de memòria de la variable en què guardarem aquesta informació. De moment, només

necessitem saber que cal posar & davant de la variable per utilitzar `scanf`.

En cas de voler llegir un caràcter després d'haver introduït un altre tipus de dada, es pot donar un comportament inadequat. Aquesta situació està causada pel salt de línia (caràcter **intro**) fet servir a l'hora de teclejar la dada prèvia, que es queda en el buffer d'entrada. Quan posteriorment fem la lectura d'un `char`, s'haurà de tenir en consideració el caràcter **intro** que hi ha al buffer d'entrada abans del caràcter introduït pel teclat. Una possible solució seria consumir abans el contingut del buffer mitjançant una crida a `getchar()`, sense assignació, i així després podrem executar la crida a `scanf` sense problemes. Per exemple:

```
scanf("%d", &i);
getchar();
scanf("%c", &c);
```

Recordeu

En introduir dades pel teclat, l'usuari ha de prémer **intro** (és a dir, introduir un caràcter de retorn de carro o el que abans hem anomenat "\n").

3. Exemples

3.1. Sortida

Exemple 03

En el programa, es defineixen dues variables, a = 2 i b = 2, i s'escriu per pantalla el missatge: «La suma d'a i b és : 4». Per tant, el resultat de sumar les dues variables és 4.

Pseudocodi	C

algorithm sumOut	`#include <stdio.h>`
	
	`int main() {`

```
var
    a: integer;
    b: integer;
end var

a := 2;
b := 2;

writeString("The result of the sum of a and b is:");
writeInteger(a + b);

end algorithm
```

```c
    int a;
    int b;

    a = 2;
    b = 2;

    printf("The result of the sum of a and b is: %d", a + b);

return 0;
}
```

3.2. Entrada i sortida de dades

Exemple 04

El programa següent rep dos valors enters pel teclat i en suma el resultat.

Pseudocodi	C

```
algorithm sum

var
    a: integer;
    b: integer;
    s: integer;
end var

writeString("Introduce a:");
a := readInteger();
writeString("Introduce b:");
b := readInteger();

s := a + b;
writeString("a + b = ");
writeInteger(s);
```

```c
#include <stdio.h>

int main() {
    int a;
    int b;
    int s;

    printf("Introduce a:");
    scanf("%d", &a);
    printf("Introduce b:");
    scanf("%d", &b);

    s = a + b;
    printf("a + b = %d \n", s);

    return 0;
}
```

end algorithm	

Exemple 05

El programa següent rep dos valors booleans pel teclat i mostra si són iguals o diferents.

Pseudocodi	C

algorithm booleanComparison
 	
  var
  b1, b2, areEquals: **boolean**;
  end var
 	
  writeString("Introduce bl: ");
  b1 := **readBoolean**();
  writeString("Introduce b2: ");
  b2 := **readBoolean**();
 	
  areEquals := b1 = b2;
  writeString("Are equals? ");
  writeBoolean(areEquals);
 	
end algorithm
	`#include <stdio.h>`
	`#include <stdbool.h>`
	
	`int main() {`
	`  bool b1, b2, areEquals;`
	`  int intToBool;`
	
	`  printf("Introduce b1 (0-FALSE, 1-TRUE): ");`
	`  scanf("%d", &intToBool);`
	`  b1 = (bool)intToBool;`
	
	`  printf("Introduce b2 (0-FALSE, 1-TRUE): ");`
	`  scanf("%d", &intToBool);`
	`  b2 = (bool)intToBool;`
	
	`  areEquals = b1 == b2;`
	`  printf("Are equals (0-FALSE, 1-TRUE)? %d\n", areEquals);`
	
	`  return 0;`
	`}`

Resum

- La sortida i l'entrada estàndard es fa per mitjà dels canals següents:
  - **Sortida**: per pantalla.

  - **Entrada**: per teclat.
- En pseudocodi, podem utilitzar una sèrie de funcions per llegir i mostrar dades pels canals estàndard.
- En C, també hi ha una sèrie de funcions per llegir i mostrar dades pels canals estàndard. Per utilitzar-les, cal afegir la línia següent en els programes C:

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

- L'ús d'aquestes funcions, tant en pseudocodi com en C, està descrit més amunt.