2. Tipus elementals de dades

1. La informació en un ordinador

Els algorismes i els programes sempre es poden pensar com un processament d'informació.

Per exemple, un programa que calculi el preu d'un pis en funció de la superfície pot prendre com a dades d'entrada la superfície del pis en metres quadrats, dades amb les quals farà els càlculs necessaris (processarà la informació) per calcular el preu del pis. Aquests càlculs poden ser més simples (podria multiplicar les dues quantitats) o més elaborats (podria tenir en compte el codi postal d'on es troba el pis per obtenir un preu per metre quadrat realista). Tanmateix, el concepte global és el mateix: tenim unes dades d'entrada i unes altres de sortida.

Aquesta estructura sempre es repeteix: s'utilitza una sèrie de dades per manipular-les o processar-les i obtenir un resultat que ens interessa.

En el cas dels càlculs, utilitzarem nombres, però també podríem parlar de text (com el mateix programa «Hello World!», en què imprimim per pantalla una frase).

Així doncs, depenent del que vulguem que faci el programa, haurem d'usar un tipus de dades o unes altres.

Però, quins són els tipus de dades que podem utilitzar en els nostres programes?

1.1. Digitalització

Heu sentit parlar de l'era digital? Es refereix, en general, a l'època en què les tecnologies de la informació tenen una incidència més rellevant per a la humanitat. Però per què es diu *digital*? La **digitalització** és el procés pel qual una quantitat es mesura i es desa com un nombre. **Digitalitzar** és transformar una quantitat física en un nombre. Per exemple, la temperatura és una magnitud de la calor de les coses. La podem mesurar usant un termòmetre i, per tant, podem assignar-hi un valor: la temperatura normalitzada és de 25 ºC. Així doncs, és relativament fàcil mesurar la temperatura i guardar-ne el valor cada hora, cada dia, o bé mesurar la temperatura mitjana de cada any.

Però, què vol dir digitalitzar una imatge? O una cançó? El procés de digitalització d'una imatge o d'una cançó és semblant al de la temperatura: utilitzem un dispositiu que és capaç de «mesurar» una imatge (com una càmera, que transforma la quantitat de llum en un senyal elèctric) o un so (com un micròfon, que transforma les vibracions generades pel so en variacions d'una ona elèctrica) i transformar el senyal que detecta en un nombre que codifica aquesta imatge o cançó. Evidentment, no és tan fàcil com mesurar la temperatura, però el procés és anàleg.

En resum, digitalitzar és transformar una quantitat física del món real en nombres.

Hi ha moltíssims perifèrics d'un ordinador que permeten transformar en nombres quantitats físiques. Hem parlat de la temperatura (termòmetre), les imatges (càmera) o els sons (micròfon). Però n'hi ha molts més:

- El camp magnètic: brúixola.
- El moviment: giroscopi o acceleròmetre.
- La posició: GPS.
- La distància: telèmetre làser.
- Composició química: espectròmetre.

Gairebé qualsevol quantitat física es pot mesurar, quantificar i codificar en un nombre. I tota

aquesta informació es pot guardar a la memòria d'un ordinador i processar-la en un programa.

1.2. Memòria d'un ordinador: sistema binari

La digitalització és molt important en les tecnologies de la informació, i també ho és el desenvolupament de les tecnologies que permeten guardar els valors de les mesures de la digitalització, això és, la **memòria de l'ordinador**.

Hi ha diferents tipus de memòries d'un ordinador, però totes es basen en el mateix concepte: la capacitat de guardar una gran quantitat de 0 i 1. No entrarem en com és la tecnologia que fa això, però la memòria RAM d'un ordinador és capaç de guardar moltíssims d'aquests 0 i 1, i llegir-los i escriure'ls de manera molt ràpida. A més, els discos durs són capaços de mantenir aquesta informació fins i tot quan no els arriba electricitat (en apagar l'ordinador).

Com s'utilitzen aquests 0 i 1?

La unitat d'emmagatzematge que pot guardar un 0 o un 1 es diu *bit*.

Fixeu-vos en què si tenim 1 bit podem comptar fins a 2: 1 i 2 (o en realitat, 0 i 1).

Si tenim 2 bits, podem comptar fins a 4:

1r bit	2n bit	Nombre
0	0	0
1	0	1
0	1	2
1	1	3

Si hi afegim un bit, podem doblar els nombres que podem codificar, i així successivament.

Nombre de bits	Nombres que podem codificar
1	2
2	$4 = 2*2 = 2^2$
3	$8 = 2*2*2 = 2^3$

4	$16 = 2*2*2*2 = 2^4$

Així doncs, si prenem cada cop més i més bits, podem codificar més i més nombres, és el que es diu *base 2*, o nombres binaris. D'aquesta manera, els nombres poden semblar molt grans: 100 en binari s'escriu 1100100, i, de vegades, resulta incòmode. Per això, i per altres raons que anirem estudiant en aquesta unitat, en ocasions es parla d'unitats de memòria més grans:

Nom	Símbol	Equivalència	Valors
Bit	bit/b		0.1
Byte	B	8 bits	0-255
Kilobyte	KB	1024 bytes (2^{10} B)	
Megabyte	MB	1024 KB (2^{20} B)	
Gigabyte	GB	1024 MB (2^{30} B)	
Terabyte	TB	1024 GB (2^{40} B)	

2. Tipus elementals de dades

Ja hem explicat més amunt que, en escriure un algorisme i codificar-lo finalment en un programa, cal seleccionar els tipus de dades adequades, és a dir, aquelles que descriuen millor les que utilitzem en la solució del problema. En aquest apartat, descriurem els diferents tipus elementals de dades que fem servir en generar un algorisme en pseudocodi i, posteriorment, codificar-lo en C.

Els tipus de dades elementals són els següents:

- Booleà
- Enter
- Real
- Caràcter
- Vector
- Cadena de caràcters (*string*)

En aquest apartat, descriurem els quatre primers, però no el cinquè ni el sisè, vector i cadena de caràcters (*string*), perquè encara no disposem dels conceptes necessaris per poder entendre'l bé. El descriurem amb detall quan parlem dels vectors, en **10. Vectors i matrius, (https://aula.uoc.edu/**

courses/78741/pages/10-vectors-i-matrius-2-2) i de les cadenes de caràcters, en **11. Tipus elementals de dades: cadenes de caràcters (https://aula.uoc.edu/courses/78741/pages/11-tipus-elementals-de-dades-cadenes-de-caracters-2-2)** . No obstant això, ens semblava necessari que, tot i que no les puguem tractar en profunditat encara, sabeu que les cadenes de caràcters (*strings*) són també un tipus de dada elemental.

Compte!

Aquí us parlarem d'uns tipus de dades que usarem en el curs, però en C n'hi ha força més, que cobreixen necessitats específiques que es plantegen en programació. A l'**enllaç, (https://en.wikipedia.org/wiki/C_data_types)** podeu veure que la llista és força més llarga del que us presentem en aquesta unitat. Si apreneu els tipus de dades que us proposem en aquesta unitat i n'enteneu bé les diferències i els usos, no us costarà entendre tots els altres. Però no cal que els aprengueu en aquest curs; simplement sigueu conscients que hi ha més tipus de dades en C.

2.1. Tipus booleà, boolean

El tipus **booleà** és el tipus de dades més simple que hi ha. Només admet dos valors: `true` (cert) o `false` (fals). Encara que no sigui estrictament correcte, de vegades també es parla d'1 o 0, respectivament. Això es fa perquè s'associa el valor 1 a `true`, i el valor 0 a `false`. Però en realitat, i encara que en aquests materials de vegades es parli de 0 i 1, les dades booleanes només poden ser `true` o `false`.

El nom d'aquestes dades, *booleanes*, procedeix de l'àlgebra de Boole, que està estretament lligada a la lògica. L'àlgebra de Boole defineix operacions lògiques entre dades de tipus booleà. És semblant a la suma i la resta que coneixem per als nombres, però «adaptada» a les dades de tipus booleà, que, com hem dit, només poden prendre els valors `true` (cert o 1) o `false` (fals o 0).

Imaginem que tenim dos valors booleans, que anomenarem a i b. Cadascun d'aquests valors pot concretar-se en `false` o `true`. Les operacions que es poden fer entre dues dades booleanes són les següents:

NOT a (operació NEGACIÓ de a)

L'operació negació s'aplica a un valor booleà, en aquest cas a. L'operació NOT (negació) canvia el valor de a:

- si a era `true` (o 1) passa a ser `false` (o 0),
- si a era `false` (o 0) passa a ser `true` (o 1).

Podem resumir l'operació NOT en la taula següent:

a	NOT a
1	0
0	1

a AND b (operació a I b)

Depenent dels valors que prenguin a i b, l'operació a AND b dona com a resultat `true` o `false`. Si a i b són `true`, l'operació a AND b (a i b) dona com a resultat `true`. D'alguna manera, ens diu que l'operació AND dona com a resultat `true` si a i b són `true`. En canvi, si una de les dues, o totes dues, són `false`, l'operació dona `false`.

Ho resumim en la taula següent:

a	b	a AND b
1	1	1
1	0	0
0	1	0
0	0	0

a OR b (operació a O b)

Semblant al cas anterior, l'operació a OR b es pot definir segons la taula següent:

a	b	a OR b
1	1	1

1	0	1
0	1	1
0	0	0

En aquest cas, el resultat de l'operació a OR b és 1 si un dels dos valors, a o b, és 1. Si tots dos són 0, llavors el resultat és 0.

Fixeu-vos en què els operadors booleans (NOT, AND i OR) estan definits només entre dades de tipus booleà i, alhora, donen com a resultat una dada de tipus booleà. Veureu per què insistim en aquest punt més endavant.

Els valors booleans també es poden comparar entre si. Si tenim dos valors booleans, a i b, podem veure si:

- a i b són iguals: =
- a i b són diferents: ≠
- a és més gran que b: >
- a és més gran que a o igual a b: ≥
- a és més petit que b: <
- a és més petit que 1 o igual a b: ≤

Si posem aquestes operacions en format de taula, com hem vist més amunt, tindrem que:

a	b	a=b	a≠b	a>b	a≥b	a<b	a≤b
0	0	1	0	0	1	0	1
0	1	0	1	0	0	1	1
1	0	0	1	1	1	0	0
1	1	1	0	0	1	0	1

Com a exercici, podríeu comprovar si la taula anterior és correcta.

2.2. Tipus booleà en C: `bool`

Recordeu que la programació implica primer dissenyar un programa utilitzant pseudocodi. En un segon pas, implementarem aquest programa; en el nostre cas, ho farem en C. En l'apartat anterior, parlàvem de la definició del tipus de dades booleanes. Ara, introduirem com aquest tipus de dades

s'implementa en C.

Quan escrivim un programa en C, tot el que hem explicat anteriorment serveix, però els noms dels operadors que hem definit canvien una mica. En C, s'ha convingut que aquests operadors s'anomenin com llistem a continuació:

Pseudocodi	C	
NOT	!	!a
AND	&&	a&&b
OR	||	a||b
=	==	a==b
≠	!=	a!=b
>	>	a>b
≥	>=	a>=b
<	<	a<b
≤	<=	a<=b

Compte!

Ho veurem amb més detall en el moment en què expliquem com es defineix una variable, a **3. Variables (i constants) i la seva definició (https://aula.uoc.edu/courses/78741/pages/3-variables-i-constants-i-la-seva-definicio-2-2)** , però en aquest moment és interessant comentar que les variables tipus `bool`, tal com les hem introduït aquí, no es consideren un tipus elemental de dades.

Per tant, quan codifiquem en C i volem usar variables `bool`, hem d'afegir la línia següent al principi del programa:

```
#include <stdbool.h>
```

2.3. Tipus enter, integer

Les dades de tipus integer són semblants als nombres enters: són els nombres 0, 1, 2, etc. I també els negatius: -1, -2, -3, etc.

Les dades de tipus integer ens permeten fer operacions entre elles. Però, a diferència de les dades de tipus booleà, les dades de tipus integer poden operar entre si mitjançant dos tipus d'operacions diferents:

- **Operacions internes**: són les operacions entre nombres enters que donen com a resultat un altre nombre enter.
- **Operacions externes**: són les que donen com a resultat un valor que no és integer. Com veurem a continuació, les operacions externes són les que comparen nombres integer entre si i donen com a resultat un valor booleà.

Els operadors interns, en el cas dels nombres enters, són els següents:

Nom	Símbol	Exemples
Canvi de signe	-	-8 -(-9)
Suma Resta	+ -	3 + 2, dona com a resultat 5. 5 - 8, dona com a resultat -3.
Multiplicació Divisió	* div	5 * 8, dona com a resultat 40. 9 div 7, dona com a resultat 1. Recordeu que el resultat d'un operador intern entre integer és un integer; per tant, el resultat no té decimals. Si utilitzeu la divisió, es pot donar el cas que el nombre pel qual es divideixi sigui 0 i es produeixi un error en el programa.
Mòdul	mod	8 mod 2 Aquesta operació retorna la resta de la divisió del primer nombre entre el segon. En aquest cas, 8 mod 2 dona com a resultat 0, mentre que 7 mod 2 dona com a resultat 1.

Els operadors externs, en el cas dels nombres integer, consisteixen en operadors de comparació, és a dir, comparen dos nombres segons una premissa:

- Són iguals?
- Són diferents?
- És el primer més gran que el segon?
- Etc.

Semblant al cas dels booleans, el resultat de comparar dos nombres integer és `true` o `false`.

Fixeu-vos-hi: és 8 diferent de 9? Cert. Tenim dos nombres enters, 8 i 9, que comparem per veure si són diferents. El resultat de l'operació no és un nombre enter, sinó booleà (veritable o cert). Per això, es diuen *operadors externs*.

Els operadors externs, en el cas dels nombres integer, són els següents:

Nom	Símbol	Exemples
Igualtat (comparació)	=	2 = 0: és 2 igual a 0? El resultat és fals. 5 = 5: és 5 igual a 5? El resultat és cert.
Desigualtat	≠	2 = 0: és 2 diferent de 0? El resultat és cert. 5 = 5: és 5 diferent de 5? El resultat és fals. Fixeu-vos en què aquests resultats i els de l'operador igualtat són justament a l'inrevés.
Més petit que Més gran que	< >	5 < 0: és 5 més petit que 0? El resultat és fals. 5 > -1: és 5 més gran que -1? El resultat és cert.
Més petit o igual que Més gran o igual que	≤ ≥	2 ≤ 0: és 2 més petit o igual que 0? El resultat és fals. 5 ≥ 5: és 5 més gran o igual que 5? El resultat és cert.

2.4. Tipus enter en C: `int`

Els nombres integer en C es diuen `int`.

Els nombres enters en matemàtiques poden prendre valors entre -∞ i +∞, però això no és possible en la realitat. Com hem comentat anteriorment, el rang de nombres enters que es pot representar

pot ser molt gran i depèn del nombre de bits que s'utilitzin per guardar un nombre enter. Com que la memòria d'un ordinador no és infinita, no podem guardar nombres amb valors que no càpiguen en aquesta memòria.

Quan es crea un compilador, es decideix que els nombres `int` ocuparan un espai de memòria determinat. Depenent dels criteris que s'adopten en crear un compilador, es decideix una cosa o una altra. Normalment, els nombres `int` ocupen entre 2 i 4 bytes.

Aquesta convenció, utilitzar 2 o 4 bytes per representar un nombre `int`, és una de les raons per les quals utilitzem una màquina virtual en aquesta assignatura. Amb la màquina virtual, ens assegurem que tots treballem «en el mateix ordinador» (com a mínim, en el mateix ordinador virtual) i tenim el mateix compilador. D'aquesta manera, no tindrem diferències pel que fa a la mida dels nombres, perquè aquesta diferència pot donar lloc a resultats diferents si no anem amb compte.

En el cas de CodeLite, en la nostra màquina virtual, els nombres `int` ocupen 4 bytes.

Bytes	1								2								3								4							
Bits	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	17	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	Signe	Nombre																														

Segons això, el rang de nombres enters que podem representar amb una dada de tipus `int` és el següent:

- Mínim: $-2^{31} = -2.147.483.648$
- Màxim: $2^{31} = 2.147.483.648$

En C, els **operadors interns** dels nombres `int` són els següents:

Nom	Símbol	Exemples
Canvi de signe	-	-8 -(-9)
Suma Resta	+ -	3 + 2, dona com a resultat 5. 5 - 8, dona com a resultat -3.
Multiplicació	*	5 * 8, dona com a resultat 40. 7 / 9, dona com a resultat 0.

Divisió	/	Recordeu que el resultat d'un operador intern entre dos nombres `int` és un `int`. Per tant, el resultat de la divisió entre `int` no té decimals. Si utilitzeu la divisió, es pot donar el cas que el nombre pel qual es divideixi sigui 0 i es produeixi un error en el programa.
Mòdul	%	8 % 2 Aquesta operació retorna la resta de la divisió del primer nombre entre el segon. En aquest cas, 8 mod 2 dona com a resultat 0, mentre que 7 mod 2 dona com a resultat 1.

En C, els **operadors externs** dels nombres `int` són els següents:

Nom	Símbol	Exemples
Igualtat (comparació)	==	2 = 0: és 2 igual a 0? El resultat és fals. 5 = 5: és 5 igual a 5? El resultat és cert.
Desigualtat	!=	2 = 0: és 2 diferent de 0? El resultat és cert. 5 = 5: és 5 diferent de 5? El resultat és fals. Fixeu-vos que aquests resultats i els de l'operador igualtat són justament a l'inrevés.
Més petit que Més gran que	< >	5 < 0: és 5 més petit que 0? El resultat és fals. 5 > -1: és 5 més gran que -1? El resultat és cert.
Més petit o igual que Més gran o igual que	<= >=	2 <= 0: és 2 més petit o igual que 0? El resultat és fals. 5 >= 5: és 5 més gran o igual que 5? El resultat és cert.

Fixeu-vos que, encara que les taules d'operadors que hem escrit per als tipus de dades integer (en pseudocodi) i les dades `int` en C semblen iguals, no ho són. Intenteu trobar quines són les diferències entre les dues taules. Això és important, perquè vol dir que la notació dels operadors en pseudocodi i en C és diferent en alguns casos.

2.5 Tipus real, real

En programar, també interessa poder representar valors numèrics que no siguin enters, sinó que tinguin decimals. Aquest és el tipus de dades anomenat real.

Quan parlem de nombres reals en matemàtiques, diem que aquests inclouen també els nombres enters. Per tant, podem pensar que són molt semblants entre si. Des d'un punt de vista informàtic, els enters i els reals són dos tipus diferents de dades i no tenen res a veure els uns amb els altres. La manera de representar-los internament és molt diferent. Això no vol dir que no puguem operar amb ells, però, com veurem més endavant, per fer-ho, caldrà convertir els valors reals en enters o al revés.

Els operadors interns i externs per als nombres reals són molt semblants als dels nombres enters.

Els operadors interns són els següents:

Nom	Símbol	Exemples
Canvi de signe	-	-8.1 -(-10.45)
Suma Resta	+ -	3.6 + 2.4, dona com a resultat 6.0. 5.0 - 8.3, dona com a resultat -3.3.
Multiplicació Divisió	* /	5.0 * 8.1, dona com a resultat 40.5. 5.0 / 2.0, dona com a resultat 2.5. Vegeu que ara no parlem de l'operador div. Aquest operador només es defineix entre enters. Per dividir reals utilitzem /. Si utilitzeu la divisió, es pot donar el cas que el nombre pel qual es divideixi sigui 0 i es produeixi un error en el programa.

Fixeu-vos que entre nombres reals no es defineix l'operador mod: entre nombres reals no hi ha una resta perquè la divisió està ben definida. Per aquesta raó, no utilitzem l'operador div, que és exclusiu dels nombres integer.

Com en el cas dels tipus de dades integer, també tenim els operadors externs, que comparen els

valors de les dades. En aquest cas, els operadors externs són els següents:

Nom	Símbol	Exemples
Igualtat (comparació)	=	2.0 = 0.0: és 2.0 igual a 0.0? El resultat és fals. 5.0 = 5.0: és 5.0 igual a 5.0? El resultat és cert.
Desigualtat	≠	2.0 = 0.0: és 2.0 diferent de 0.0? El resultat és cert. 5.0 = 5.0: és 5.0 diferent de 5.0? El resultat és fals. Fixeu-vos que aquests resultats i els de l'operador igualtat són justament a l'inrevés.
Més petit que Més gran que	< >	5.0 < 0.0: és 5.0 més petit que 0.0? El resultat és fals. 5.0 > -1.0: és 5.0 més gran que -1.0? El resultat és cert.
Més petit o igual que Més gran o igual que	≤ ≥	2.0 ≤ 0.0: és 2.0 més petit o igual que 0.0? El resultat és fals. 5.0 ≥ 5.0: és 5.0 més gran o igual que 5.0? El resultat és cert.

2.6 Tipus real en C: `float`

Com en el cas dels nombres enters, hi ha infinits nombres reals, però el nostre ordinador no els pot representar tots. Respecte dels nombres enters, hem comentat que hi ha infinits nombres, tant positius com negatius. En el cas dels nombres reals, la cosa es complica encara més, ja que hi ha infinits nombres, fins i tot entre dos nombres enters. És a dir, hi ha infinits nombres entre el 0 i l'1, entre el 0 i el -1, etc.

Amb un ordinador no podem representar tots aquests nombres, per tant, necessitem representar-los d'una manera que sigui eficient. Per a això, dividim els nombres reals en dues parts:

- **Les xifres significatives (també anomenades** *mantissa* **o** *significant***)**: aquelles que aporten informació. Pensem, per exemple, en el nombre pi, π. És un nombre que té infinits decimals. En qualsevol cas, quan l'utilitzem en càlculs reals, no emprem infinites xifres, sinó que en solem usar unes quatre o cinc. Això és perquè les quantitats reals solen tenir un error associat, i el nombre de xifres que realment aporten informació és relativament petit. Per tant, no té gaire

sentit treballar amb el número 8.5278092812034780981 2734, i a més cometrem un error força petit si treballem amb 8.5278.

- **L'ordre de magnitud**: si decidim utilitzar tres xifres significatives per descriure un nombre, aquestes xifres poden estar en les unitats (per exemple, 1.23), en els milers (1230) o en les centèsimes (0.0123). Per treballar de manera eficaç amb aquestes xifres, podem utilitzar el format exp o potència de 10. Així, els nombres anteriors es poden escriure com:
  - $1.23 = 1.23 \times 1 = 1.23 \times 10^0 = 1.23E0$
  - $1230 = 1.23 \times 1000 = 1.23 \times 10^3 = 1.23E3$
  - $0.0123 = 1.23 \times 0.01 = 1.23 \times 10^{-2} = 1.23E-2$

Fixeu-vos que, com en els exemples anteriors, tots els nombres tenen el mateix nombre de xifres significatives (tres), però amb ordres de magnitud diferents.

Així, els nombres molt grans o molt petits es poden representar d'una manera més eficient:

- 0.000000000000000000000000000000009235845845 = 9.235845845E-35
- 2558846522018200000000000 = 2.5588465220182E24

La denominació `float` es refereix a la coma flotant, és a dir, la coma denota la posició de la primera xifra significativa i pot estar en la magnitud unitats, centèsimes o desenes de milers.

En el cas de CodeLite, els nombres `float` estan codificats en 4 bytes o, cosa que és el mateix, 4 x 8 = 32 bits. La codificació d'aquests nombres es fa de la manera següent:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	17	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Signe	Magnitud (exponent)								Xifres significatives																						

És una mica més complicat de calcular que en el cas de les dades `int`, però es pot concloure el següent:

- Els valors dels tipus de dades `float` (codificats en 4 bits) van de -3.4×10^{38} a $+3.4 \times 10^{38}$.
- El valor `float` positiu més petit i diferent de 0 que es pot representar és 1.7×10^{-38}.
- Els tipus de dades `float` solen tenir unes 6-7 xifres significatives.

Els operadors externs de les dades de tipus `float` són els següents:

Nom	Símbol	Exemples
Igualtat (comparació)	==	2.0 = 0.0: és 2.0 igual a 0.0? El resultat és fals. 5.0 = 5.0: és 5.0 igual a 5.0? El resultat és cert.
Desigualtat	!=	2.0 = 0.0: és 2.0 diferent de 0.0? El resultat és cert. 5.0 = 5.0: és 5.0 diferent de 5.0? El resultat és fals. Fixeu-vos que aquests resultats i els de l'operador igualtat són justament a l'inrevés.
Més petit que Més gran que	< >	5.0 < 0.0: és 5.0 més petit que 0.0? El resultat és fals. 5.0 > -1.0: és 5.0 més gran que -1.0? El resultat és cert.
Més petit o igual que Més gran o igual que	<= >=	2.0 ≤ 0.0: és 2.0 més petit o igual que 0.0? El resultat és fals. 5.0 ≥ 5.0: és 5.0 més gran o igual que 5.0? El resultat és cert.

Fixeu-vos que els operadors externs del tipus de dades `float` són els mateixos que els del tipus de dades `int`.

2.7. Tipus caràcter, char

Un altre tipus de dada és el tipus caràcter, char, que serveix per representar:

- Lletres.
- Dígits (no nombres, sinó dígits, és a dir, els caràcters que representen nombres).
- Símbols de puntuació.
- Altres símbols utilitzats normalment en informàtica.

Internament, els caràcters estan digitalitzats i representats per nombres. A cada caràcter li correspon un nombre. Per exemple, podríem inventar una representació de la manera següent:

- Els dígits del 0-9 podrien tenir els codis 0-9.
- Les lletres de l'abecedari (a, b, ..., z) podrien tenir els codis des de 10 a 26.

Però encara ens faltarien les lletres accentuades (á, à, é, è, etc.), les dièresis (ï, ü, etc.), els signes d'admiració, els interrogants, etc. És més complicat del que sembla, i hi ha moltes maneres diferents de fer-ho.

El conjunt de correspondències entre caràcters, nombres i símbols es diu **codi**.

Hi ha diferents codis, però el més utilitzat i el que s'ha convertit en l'estàndard és el **codi ASCII** (*American Standard Code for Information Interchange*), que utilitza un byte per representar cada caràcter. Per tant, en el codi ASCII es poden representar fins a 256 caràcters diferents. Trobareu la llista completa de **codis ASCII (https://en.wikipedia.org/wiki/ASCII)** fàcilment.

No hi ha una manera d'operar caràcters entre si: no podem calcular «a + b» ni «? * A». Això no té sentit. Per tant, **no hi ha operadors interns entre dades de tipus caràcter**.

En canvi, atès que els codis (com el codi ASCII) assignen un nombre a cada caràcter, sí que es pot definir un ordre entre dades `char` i fer operacions de comparació entre ells. Recordeu que el codi ASCII assegura que les lletres estan ordenades alfabèticament, però cal tenir en compte que les majúscules i les minúscules no estan les unes a continuació de les altres.

Nom	Símbol	Exemples
Igualtat (comparació)	=	a = a: el resultat és cert. A = ?: el resultat és fals.
Desigualtat	≠	a = a: el resultat és fals. A = ?: el resultat és cert.
Més petit que Més gran que	< >	A < b: el resultat és cert. B < a: el resultat és cert. B > A: el resultat és cert. B > a: el resultat és cert. Reviseu la **taula del codi ASCII (https://en.wikipedia.org/wiki/ASCII)** per comprovar aquests resultats.
Més petit o igual que Més gran o igual	≤ ≥	A ≤ b: el resultat és cert. B ≤ B: el resultat és cert.

que		B ≧ A: el resultat és cert. a ≧ a: el resultat és cert.

Una dada de tipus caràcter només guarda un únic caràcter. Més endavant, ja veurem com es poden representar cadenes de caràcters per poder desar paraules o frases completes.

2.8. Tipus caràcter en C: `char`

El que hem explicat per als tipus de dada `char` en l'apartat anterior serveix també per a les dades tipus `char` en C. L'únic que és diferent en aquest cas és que els operadors externs canvien de la mateixa manera que en les dades de tipus `int` o `float`:

Nom	Símbol	Exemples
Igualtat (comparació)	==	a = a: el resultat és cert. A = ?: el resultat és fals.
Desigualtat	!=	a = a: el resultat és fals. A = ?: el resultat és cert.
Més petit que Més gran que	< >	A < b: el resultat és cert. B < a: el resultat és cert. B > A: el resultat és cert. B > a: el resultat és cert. Reviseu la **taula del codi ASCII** (https://en.wikipedia.org/wiki/ASCII) per comprovar aquests resultats.
Més petit o igual que Més gran o igual que	<= >=	A ≦ b: el resultat és cert. B ≦ B: el resultat és cert. B ≧ A: el resultat és cert. a ≧ a: el resultat és cert.

Resum

- La informació en un ordinador està **digitalitzada**, la qual cosa ens permet desar-la de manera efectiva.
- La informació en un ordinador es desa a la **memòria** en forma de **bits** (0 i 1) que s'ajunten per formar bytes, que alhora s'uneixen per formar kilobytes, i així successivament.
- Un ordinador pot guardar diferents tipus de dades. N'hi ha molts més, però nosaltres treballarem amb:
  - **Booleanes**: aquest tipus de dada pot tenir només dos valors, cert o fals. De vegades, també es parla de 0 o 1.
  - **Enters**: es corresponen amb els nombres enters en matemàtiques.
  - **Reals** (o de coma flotant): es corresponen amb els nombres reals en matemàtiques.
  - **Caràcters**: es corresponen amb lletres, nombres i símbols. Aquests caràcters alfanumèrics es desen com a nombres a l'ordinador, de manera que la correspondència entre el nombre i el caràcter és definida per una taula (en el nostre cas, ASCII).
- Aquests tipus de dades tenen la seva correspondència en pseudocodi i codi C:

Pseudocodi	C
boolean	`bool`
integer	`int`
real	`float`
char	`char`

- Hi ha una sèrie d'operacions definides per a cadascun d'aquests tipus de dades, que donen com a resultat un nou valor del mateix tipus de dada (**operacions internes**), i operacions entre aquests tipus de dades, que donen com a resultat una variable booleana (**operacions externes**).