Pràctiques Integrades
1er de Matemàtiques

Pràctica 6

curs 2002–03

6 Llistes, conjunts i seqüències

El programa Maple permet treballar amb diferents *conjunts* o estructures de dades. Aquests *conjunts* es poden definir de diverses maneres i a més en podem canviar l'estructura d'un tipus a un altre segons ens interessi. Les estructures de dades que analitzarem en aquesta pràctica són les llistes, conjunts i seqüències.

6.1 Llistes

Una llista és un conjunt ordenat d'expressions on es permeten repeticions. L'entrada al Maple es fa entre claudàtors (`[ ]`) i separant els elements que conté amb comes.

Exemple 6.1

Definim la variable a com la llista següent $[1, 2, 3, 2, 1, 2]$:

```
> a:=[1,2,3,2,1,2];
```

Els elements d'una llista es poden cridar o recuperar un a un mitjançant la posició que ocupen escrivint aquesta entre claudàtors. Una particularitat d'aquest mètode és que quan utilitzem nombres negatius retorna l'element corresponent començant per la cua.

Exemple 6.2

Si volem recuperar o extreure el tercer element de la llista `a` escriurem:

```
> a[3];
```

Mentre que si volem l'últim podem fer-ho amb la comanda següent:

```
> a[-1];
```

De la manera semblant també podem extreure una subllista de la llista original. Si volem una subllista amb els elements entre les posicions i i j cal introduir l'argument `i..j` dins els claudàtors.

Exemple 6.3

En aquest exemple recuperarem la subllista que conté els elements des de la posició 2 fins a la 4.

```
> a[2..4];
```

Una manera de construir llistes a partir d'altres és mitjançant la unió, és a dir, també podem construir una llista com la unió de vàries. Per a això necessitem abans "*treure*" els claudàtors de les llistes inicials i afegir-los després a la unió o concatenació obtinguda. La comanda que ens permet treure els claudàtors és la comanda `op( )`. Un cop hem tret els claudàtors podem concatenar dues llistes. Per exemple, observeu el funcionament de la comanda `op` executant la següent línia de Maple.

```
> op(a);
```

Exercici 6.1

Definiu la funció `concatenar`, depenent de dues variables, que retorni la unió de dues llistes donades. Comproveu que la definició que heu fet és correcta fent la prova amb les llistes $a = [1, 2, 3, 2, 1]$ i $b = [3, 2, 4]$. El resultat ha de ser la llista $c = [1, 2, 3, 2, 1, 3, 2, 4]$.

6.2 Conjunts

En un conjunt de dades no hi ha repeticions i les dades no tenen un ordre prefixat (és a dir, Maple pot variar l'ordre, en funció del context). L'entrada a Maple es fa entre claus ({ }) i separant els elements que conté per comes. La majoria d'opcions que hem vist per les llistes són també vàlides per als conjunts, però tenint en compte les possibles diferències.

Exemple 6.4

Definiu b com el conjunt $\{1, 2, 5, 1, 3\}$ i observeu la sortida que us torna.

```
> b:={1,2,5,1,3};
```

Els elements d'un conjunt també es poden cridar mitjançant els claudàtors, tenint en compte que, a priori, no coneixem l'ordre en que Maple guarda els elements ja que són estructures no ordenades de dades.

Exemple 6.5

Per a cridar el tercer element del conjunt b fem:

```
> b[3];
```

Observeu que el tercer element no coincideix amb el que hem introduït inicialment en la posició 3 quan hem definit b.

Exercici 6.2

Definiu a i b com els conjunts $\{x, y, z\}$ i $\{t, u\}$. Definiu la unió d'aquests dos conjunts $c = a \cup b$. Construïu una funció que, a partir de dos conjunts, doni com a resultat la seva unió. Proveu-la amb els conjunts $A = \{1, 4, 9, 16\}$ i $B = \{2, 4, 6, 8, 10\}$.

6.3 Seqüències

Una seqüència d'elements és un conjunt ordenat però que s'interpreta com n arguments (i no una sola llista), on n és el nombre d'elements de la seqüència. L'entrada al Maple es realitza amb les dades separades entre comes i sense cap delimitador.

Exemple 6.6

Si volem definir com a la successió $1, 4, 9, 16$ posem

```
> a:=1,4,9,16;
```

Una altra manera d'aconseguir seqüències és mitjançant la comanda `seq` que ja va ser introduïda a la primera pràctica i vàreu utilitzar en alguns exemples.

Exemple 6.7

La seqüència c amb els primers 10 quadrats enters es construeix amb la comanda `seq` de la manera següent.

```
> c:=seq(i^2,i=1..10);
```

Igual que fèiem en el cas de les llistes o conjunts podem extreure o recuperar un element d'una seqüència, unir seqüències, ...

Exercici 6.3

Definiu la funció de dues variables $f(x, y) = x^2 - y^2$. Considereu $a = (5, 7)$. Com s'ha d'introduir a per aconseguir que en escriure $f(a)$ Maple doni com a resultat el valor -24?

6.4 Taules

Totes les estructures per emmagatzemar dades que hem vist fins ara admeten només una indexació mitjançant nombres enters. Maple té una estructura de dades més sofisticada que admet com a conjunt de indexació pràcticament qualsevol cosa. És l'estructura anomenada `table`. Com veurem més endavant fins i tot admet cadenes de caràcters com a índexs. La millor manera de veure com funciona es mitjançant uns exemples. Ja sabeu que podeu aprendre més de les seves propietats utilitzant l'ajuda de Maple.

Exemple 6.8

En aquest exemple veurem com es defineix aquesta estructura i el seu comportament respecte assignació de valors.

```
> T:=table([1=1,2=3,3=5,x=u,y=v,z=w,t=sin(2*t)]);
> T[1];
> T[2];
> T[x];
> T[5];
> t:=5;
> T[t];
> T[5];
> subs(u=3,T[x]);
```

I podem afegir-hi nous elements,

```
> T[5]:=m;
> T[5];
```

Com heu pogut observar el conjunt de índexs és qualsevol cosa i per tant, no és immediat saber el que s'ha de posar entre claudàtors per a recuperar cada un dels elements d'una estructura `table`. De la feina de recuperar el conjunt d'índexs per als que s'ha introduït algun valor en una taula se n'encarrega la comanda `indices`.

Exemple 6.9

```
> indices(T);
```

Exemple 6.10

En aquest exemple utilitzarem el format `table` d'estructura de dades per crear un petit diccionari.

```
> Traductor:=table();
> Traductor[dilluns]:=Monday;
> Traductor[dimarts]:=Tuesday;
> Traductor[dimecres]:=Wednesday;
> Traductor[dijous]:=Thursday;
> Traductor[divendres]:=Friday;
> Traductor[dissabte]:=Saturday;
> Traductor[diumenge]:=Sunday;
> Traductor[dimecres];
> Traductor[divendres];
```

Fixeu-vos que la taula `Traductor` s'ha definit inicialment sense cap assignació de valors i que cada un dels valors s'ha afegit en les comandes posteriors.

6.5 Canvis de format entre llistes, seqüències i conjunts. La comanda `convert`

Maple permet transformar una expressió que té guardada com una llista, seqüència o conjunt als altres formats.

Un d'aquests canvis ja l'hem utilitzat. A l'apartat corresponent a les llistes ja hem vist com podem canviar una llista de dades en una seqüència, mitjançant la comanda `op( )`.

Exemple 6.11

Convertirem la llista $a = [1, 2, 3, 2, 1]$ en una seqüència b.

```
> a:=[1,2,3,2,1];
> b:=op(a);
```

Un cop tenim una seqüència podem transformar-la un altre cop en una llista simplement afegint els delimitadors que caracteritzen les llistes: els claudàtors.

Exemple 6.12

Si volem transformar la seqüència b en una llista només cal afegir els claudàtors:

```
> [b];
```

El procediment anàleg funciona amb els conjunts i les seqüències. Recordeu que els delimitadors que caracteritzen els conjunts són les claus. És molt important que tingueu en compte que quan passem una seqüència o una llista a format conjunt perdem les repeticions i l'ordre que tenien originalment.

Exercici 6.4

Convertiu la llista $a = [3, 2, 1, 4, 2, 2, 2]$ en un conjunt b i feu una nova llista c amb els elements de b.

Una altra manera de fer conversions entre llistes, seqüències i conjunts és amb la comanda `convert( )`. Aquesta comanda és molt versàtil i s'utilitza també per a canvis en altres contexts dins de Maple (taules a matrius, ...).

Per a passar d'un format a l'altre tan sols cal conèixer la terminologia de Maple corresponent a cada un d'ells: `set` per a conjunts i `list` per a llistes. Per seguretat escriurem aquestes paraules entre comes ja que si haguessin estat definides com a variables serien substituides pel seu valor i així només són una paraula.

Exemple 6.13

Considerem la llista a formada pels elements $a = [3, 2, 1, 4, 3]$. Podem passar-la a format conjunt mitjançant:

```
> a:=[3,2,1,4,3];
> b:=convert(a,'set');
```

I podem tornar al format llista mitjançant:

```
> convert(b,'list');
```

6.6 Comptar elements de llistes i conjunts

La funció que permet comptar el nombre d'elements d'una llista o conjunt és la funció `nops( )`.

Exemple 6.14

Volem saber quants elements diferents té la llista $a = [3, 4, 2, 1, 3, 5, 3, 4, 3, 4, 5, 3, 2, 2, 4]$. La funció `nops( )` aplicada directament a la llista

```
> a:=[3,4,2,1,3,5,3,4,3,4,5,3,2,2,4];
> nops(a);
```

retorna el nombre d'elements de la llista (amb repeticions incloses). Una manera de saber quants elements diferents té és convertir primer la llista a conjunt b i seguidament comptar el nombre d'elements del conjunt:

```
> b:=convert(a,'set'); nops(b);
```

Exercici 6.5

Quants elements té la llista `a:=[[1,2,3],[x,y],{2, 3,4}, x, 1, 2]`? Raoneu la resposta.

6.7 Exemple: grups de permutacions

Recordeu que una permutació d'un conjunt de n elements $\{1, 2, \ldots, n\}$ és una aplicació bijectiva:

$$\begin{array}{rccc} \sigma: & \{1,2,\ldots,n\} & \longrightarrow & \{1,2,\ldots,n\} \\ & i & \mapsto & \sigma(i) \end{array}$$

El conjunt de totes aquestes permutacions es designa per S_n.

Les permutacions s'acostumen a escriure mitjançant la següent notació:

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \sigma(1) & \sigma(2) & \sigma(3) & \cdots & \sigma(n) \end{pmatrix}$$

o bé mitjançant un producte de cicles disjunts.

Per a treballar amb permutacions utilitzant Maple necessitem carregar el paquet **group** que conté comandes específiques per a la seva manipulació.

```
> with(group);
```

6.7.1 Entrar elements

Inicialment, la manera més immediata d'introduir una permutació en Maple és mitjançant les imatges de cada posició posades en una llista de la forma `[`$\sigma(1), \ldots, \sigma(n)$`]`. Ara bé, per la majoria de funcions és necessari escriure-la en la seva descomposició en cicles disjunts i això es fa introduint una *llista de llistes* de la forma `[`$c_1, \ldots, c_k$`]`, on cada c_i és una llista que representa un cicle. En particular l'element neutre (la permutació identitat) es representa en aquest context per una llista buida (`[ ]`).

Exemple 6.15

Per a entrar la permutació $\sigma = (1,2,3)(4,5)$ (producte dels cicles $(1,2,3)$ i $(4,5)$) del grup simètric S_5 definim

```
> sigma:=[[1,2,3],[4,5]];
```

En determinats casos la manera com tindrem definida una permutació serà com una llista ordenada amb les imatges de cada posició, o sigui, `[`$\sigma(1), \sigma(2), \ldots, \sigma(n)$`]`. Tot i això, si volem operar amb ella haurem de transformar-la a un producte de cicles disjunts mitjançant la comanda `convert`. En el següent exemple veurem com passar d'un format a un altre.

Exemple 6.16

Si considerem la permutació $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 3 & 5 \end{pmatrix}$ i la volem introduir amb aquesta notació escriurem,

```
> sigma:=[2,4,1,3,5];
```

Si la volem descriure com a producte de cicles disjunts utilitzarem la comanda `convert` de la manera següent,

```
> convert(sigma,'disjcyc');
```

Tenint en compte que sempre podem recuperar la notació inicial utilitzant també la comanda `convert`.

```
> convert(%,'permlist',5);
```

on l'argument 5 és per a concretar que és un element de S_5.

Exercici 6.6

Donada la permutació `tau` expressada com a producte dels cicles disjunts `[[1,3,5],[2,7,6]]`, podeu preveure quina diferència hi ha entre els resultats que s'obtenen quan executeu les comandes `convert(tau, 'permlist', 7)` i `convert(tau, 'permlist', 10)`? I si intenteu fer `convert(tau, 'permlist', 5)`?

6.7.2 Operacions amb permutacions

Les permutacions es poden multiplicar (composar) i invertir. En aquest apartat veurem com realitzar aquestes operacions usant comandes del paquet **groups** que hem carregat.

Per a multiplicar elements utilitzem la funció `mulperms( )`.

Exemple 6.17

Per a multiplicar les permutacions $\sigma_1 = (1,2,3)(4,5)$ amb $\sigma_2 = (3,4)$ escriuirem:

```
> sigma1:=[[1,2,3],[4,5]];
> sigma2:=[[3,4]];
> mulperms(sigma1,sigma2);
```

Què ha calculat: $\sigma_1 \circ \sigma_2$ o bé $\sigma_2 \circ \sigma_1$?

Per a calcular la inversa d'una permutació s'utilitza la funció `invperm( )`.

Exemple 6.18

Calculem la inversa de la permutació $(1,2,3)(5,6,4)$:

```
> sigma:=[[1,2,3],[6,5,4]];
> beta:=invperm(sigma);
```

Podem comprovar que `beta` és la inversa de `sigma`.

```
> mulperms(sigma,beta);
> mulperms(beta,sigma);
```

Exercici 6.7

Considereu les permutacions següents de S_8: $\sigma_1 = (1,2,3)(6,7,8)$, $\sigma_2 = (5,4,3,1)$ i $\sigma_3 = (2,3,4,5,6,7)$. Calculeu les composicions $\sigma_1 \circ \sigma_2 \circ \sigma_3$, $\sigma_3 \circ \sigma_2 \circ \sigma_1$, $\sigma_1^{-1} \circ \sigma_2^{-1} \circ \sigma_3^{-1}$, $\sigma_3^{-1} \circ \sigma_2^{-1} \circ \sigma_1^{-1}$, $(\sigma_1 \circ \sigma_2 \circ \sigma_3)^{-1}$ i $(\sigma_3 \circ \sigma_2 \circ \sigma_1)^{-1}$.

Quines relacions veieu entre els resultats que heu obtingut? Recordeu que el producte $\sigma_2 \circ \sigma_1$ correspon a la comanda de Maple `multperms(sigma1,sigma2)`.

6.7.3 Signe d'una permutació

Maple té implementada la funció signe d'una permutació amb el nom de `parity( )`. L'argument és una permutació que ha d'estar definida com a producte de cicles disjunts i aleshores Maple ens retorna els valors 1 o -1 depenent del signe de la permutació.

Exemple 6.19

Per calcular el signe de les permutacions $(2,1,3,5)(4,9,10)$ i $(3,4,1)(5,6,7)$ amb Maple ho faríem de la manera següent:

```
> parity([[2,1,3,5],[4,9,10]]);
> parity([[3,4,1],[5,6,7]]);
```

Exercici 6.8

Considereu la permutació de S_{10} donada per $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 6 & 9 & 10 & 4 & 7 & 8 & 2 & 1 & 5 \end{pmatrix}$. Determineu el seu signe.

6.7.4 Ordre d'una permutació

Recordeu que l'ordre d'una permutació σ és el més petit nombre natural n tal que $\sigma^n = \sigma \circ \overset{(n)}{\ldots} \circ \sigma$ és la permutació identitat. Per tant, si es vol calcular aquest ordre es pot anar calculant les potències successives de σ fins a obtenir per primer cop la identitat. Naturalment, anar fent `mulperms(mulperms(...(`σ,σ`)),`σ`)` no és gaire pràctic. Aquest procés es pot simplificar si es defineix prèviament una funció `ms( )` que multipliqui qualsevol permutació per σ, d'aquesta manera només s'ha de fer `ms(ms(ms...(`σ`)))`. Maple preveu poder fer aquest tipus d'operació d'una manera ràpida utilitzant l'operador d'iteració `@@` que permet escriure `(ms@@n)(`σ`)`, on `n` és el nombre de cops que apliquem `ms`, per a abreujar l'operació anterior.

Exercici 6.9

Donada la permutació $\sigma = (2, 3, 5, 4, 6, 1, 8, 7)$ de S_8, definiu la funció `ms` que aplicada sobre una permutació qualsevol τ doni com a resultat la permutació producte de σ per τ.

Utilitzant la funció `ms` i l'operador `@@` determineu l'ordre (i totes les potències diferents) de σ.