Programmation Logique

Entrée et sortie

Enseignant: NGUYEN Thi Minh Tuyen

Termes de sortie

write/1

- Provoque l'écriture du terme dans le flux de sortie actuel, par défaut, c'est l'écran de l'utilisateur.

```
?- write(26),nl.
26
true.
?- write('a string of characters'),nl.
a string of characters
true.
?- write([a,b,c,d,[x,y,z]]),nl.
[a,b,c,d,[x,y,z]]
true.
?- write(mypred(a,b,c)),nl.
mypred(a,b,c)
true.
```

Termes de sortie

writeq/1

- La sortie avec les guillemets

```
?- writeq('a string of characters'),nl.
'a string of characters'
true.
?-writeq(dog),nl.
dog
true.
?- writeq('dog'),nl.
dog
true.
```

Termes d'entrée

read/1

- Provoque la lecture du terme à partir du flux d'entrée actuel, qui est par défaut le clavier de l'utilisateur.

```
?- read(X).
|:jim.
X = jim
?- read(X).
|:26.
X = 26
```

```
?- read(X).
|:mypred(a,b,c).
X = mypred(a,b,c)
?- read(Z).
|: [a,b,mypred(p,q,r),[z,y,x]].
Z = [a,b,mypred(p,q,r),[z,y,x]]
?- read(Y).
|: 'a string of characters'.
Y = 'a string of characters'
?- X=fred,read(X).
|:jim.
false.
?- X=fred,read(X).
|:fred.
X = fred
```

Entrée et sortie en utilisant des caractères

9	tab	40	(	59	;	94	^
10	end of record	41	)	60	<	95	_
32	space	42	*	61	=	96	`
33	!	43	+	62	>	97–122	a to z
34	"	44	,	63	?		
35	#	45	-	64	@	123	{
36	$	46	.	65-90	A to Z	124	|
37	%	47	/	91	[	125	}
38	&	48-57	0 to 9	92	\	126	~
39	'	58	:	93	]		

Caractères de sortie

put/1

- Le prédicat prend un argument, qui doit être un entier entre 0 et 255.
- L'évaluation du prédicat: provoque l'affichage du caractère corespondant à la valeur ASCII.

```
?- put(97),nl.
a
true.
?- put(122),nl.
z
true.
```

Caractères d'entrée

`get0/1 et get/1`

- Prend une variable.
- L'évaluation de `get0` provoque la lecture d'un caractère dans le flux d'entrée actuel.
  - La variable est alors unifiée avec la valeur ASCII de ce caractère.
- L'évaluation de `get` provoque la lecture le prochain caractère non-d'espace à partir du flux d'entrée actuel.
  - La variable est alors unifiée avec la valeur ASCII de ce caractère.

Exemple


```
?- get0(N).
|: a
N = 97
?- get0(N).
|: Z
N = 90
?- get0(M)
|:)
M = 41
?- M is 41,get0(M).
|:)
M = 41


?- M is 50,get0(M).
|:)
false.
?- get(X).
|: Z
X = 90
?- get(M).
|: Z
M = 90
```

Exemple

- Lisez une chaîne de caractères (se termine par *) et affichez la valeur ASCII de chaque caractères à l'écran.

```
readin:-get0(X),process(X).
process(42).
process(X):-X=\=42,write(X),nl,readin.
```

Exercice: Majuscule → minuscule

9	tab	40	(	59	;	94	^
10	end of record	41	)	60	<	95	_
32	space	42	*	61	=	96	`
33	!	43	+	62	>	97–122	a to z
34	"	44	,	63	?		
35	#	45	-	64	@	123	{
36	$	46	.	65-90	A to Z	124	|
37	%	47	/	91	[	125	}
38	&	48-57	0 to 9	92	\	126	~
39	'	58	:	93	]		

Exercice: Majuscule → minuscule

Entrée et sortie à l'aide de fichiers

Fichier de sortie: Modification du flux de sortie actuel

- `tell/1`
  - Le fichier nommé dans l'argument devient le flux de sortie actuel. Si le fichier n'est pas déjà ouvert, un fichier portant le nom spécifié est d'abord créé (tout fichier existant portant le même nom est supprimé).
  - Exemple: `tell('outfile.txt').`
- `told/0`
  - Entraîne la fermeture du fichier de sortie actuel et réinitialise le flux de sortie actuel en utilisateur – le terminal de l'utilisateur.
- `telling/1`
  - L'argument doit être une variable, qui est liée au nom du flux de sortie actuel.

Entrée de fichier: modification du flux d'entrée actuel

- `see/1`
  - Le fichier spécifié dans l'argument devient le flux d'entrée actuel. Si le fichier n'est pas déjà ouvert, il est d'abord ouvert. S'il n'est pas possible d'ouvrir un fichier avec le nom donné, une erreur sera générée.
  - Exemple: `see('myfile.txt')`
- `seen/0`
  - Entraîne la fermeture du fichier d'entrée actuel et réinitialise du flux d'entrée actuel en utilisateur – le terminal de l'utilisateur.
- `seeing/1`
  - L'argument doit être une variable, qui est liée au nom du flux d'entrée actuel.

Lire une ligne


```
readline:-get0(X),process(X).
process(10).
process(X):-X=\=10,put(X),nl,readline.
```

Exemple

1. Lisez et affichez le contenu d'un fichier.
2. Lisez d'un fichier, copiez le contenu et l'écrivez dans un autre fichier.

Lecture à partir des fichiers

- Dans Prolog, la lecture des informations à partir de fichiers est
  - simple si les informations sont données sous la forme de termes Prolog suivis par des terminaux.
  - plus difficile si le format des données ne sont pas données.
- Nous pouvons utiliser les flux et deux prédicats `open` et `close`.

Exercice: copie des termes

- Entrée:

```
'first term'. 'second term'.
'third term'.
fourth. 'fifth term'.
sixth.
```

- Sortie:

```
'first term'.
'second term'.
'third term'.
fourth.
'fifth term'.
sixth.
```

```
copyterms(Infile,Outfile):-
        seeing(S),telling(T),
        see(Infile),tell(Outfile),
        copy,
        seen,see(S),told,tell(T).
copy:-read(X),process(X).
process(end_of_file).
process(X):- X \= end_of_file,
             writeq(X),write('.'),nl,
             copy.
```

Exemple

- *houses.txt:*

gryffindor. hufflepuff. ravenclaw. slytherin.

```
main:- open('houses.txt',read,S),
       read(S,H1),
       read(S,H2),
       read(S,H3),
       read(S,H4),
       close(S),
       write([H1,H2,H3,H4]), nl.
```

Atteindre la fin d'un flux

`at_end_of_stream/1`

- vérifie si la fin d'un flux est atteinte

```
main:-   open('houses.txt',read,Str),
         read_houses(Str,Houses),
         close(Str),
         write(Houses),  nl.

read_houses(Stream,[]):-
       at_end_of_stream(Stream).

read_houses(Stream,[X|L]):-
         \+ at_end_of_stream(Stream),
         read(Stream,X),
         read_houses(Stream,L).
```

Lecture des entrées arbitraires

`get_code/2`

- Lit le prochain caractère disponible du flux
  - Premier argument: un flux
  - Deuxième argument: le code du caractère

Exemple


```
readWord(Stream,Word):-
        get_code(Stream,Char),
        checkCharAndReadRest(Char,Chars,Stream),
        atom_codes(Word,Chars).

checkCharAndReadRest(10, [], _):- !.
checkCharAndReadRest(32, [], _):- !.
checkCharAndReadRest(-1, [], _):- !.
checkCharAndReadRest(Char,[Char|Chars],S):-
        get_code(S,NextChar),
        checkCharAndRest(NextChar,Chars,S).
```

Exercice

1. Écrivez un programme Prolog qui permet de créer un fichier `sortieavecformat.txt` contenant les lignes suivantes:

```
          un
    deux        trois
quatre      cinq        six
```

Vous pouvez utiliser ces prédicats pré-definis: open/3 , close/1 , tab/2 , nl/1 , et write/2.

2. Écrivez un programme Prolog qui lit dans un fichier texte mot par mot, affiche chaque mot sur une ligne.
3. Écrivez un programme Prolog qui lit dans un fichier texte mot par mot, et affiche tous les mots lus et leur fréquence (sépérateurs: espace, virgule, point)

Question?