Initiation à la programmation

L'unité cartes

Présentation

L'unité `cartes` offre une extension du langage PASCAL afin d'écrire des programmes destinés à un robot manipulateur de cartes.

Le domaine de travail du robot est constitué de 4 tas, numérotés 1,2,3,4, sur lesquels sont empilées des cartes à jouer. Ces cartes ont les couleurs habituelles (♣♢♠♡) et les valeurs habituelles (par ordre croissant : as, deux, trois,..., dix, valet, dame, roi). Les cartes provenant de plusieurs jeux, il est possible de trouver plusieurs exemplaires d'une figure donnée dans une couleur donnée.

Chacun des quatre tas peut contenir un nombre quelconque de cartes, y compris aucune.

Le langage de l'unité `cartes`

Description des tas

Afin de décrire en début de programme quelle est la configuration initiale des tas de cartes, l'unité `cartes` offre une procédure `InitTas` à deux paramètres qui permet de décrire un tas :

`InitTas(n,s)`

où `n` est le numéro du tas décrit, et `s` est une *chaîne de description* du tas.

La chaîne de description, lue de gauche à droite, indique les cartes d'un tas du bas vers le haut, avec les conventions suivantes :

- Les lettres `T, K, P, C` représentent respectivement ♣, ♢, ♠, ♡
- Si `A` et `B` sont des chaînes de description, la chaîne `AB` est aussi une chaîne de description qui représente les cartes de `A` surmontées de celles de `B`.
- Si `A` et `B` sont des chaînes de description, la chaîne `A+B` est aussi une chaîne de description qui repésente un choix entre les cartes de `A` ou celles de `B`
- Si `A` est une chaîne de description, `[A]` représente la répétition des cartes de `A` un nombre quelconque (défini de manière aléatoire) de fois (y compris zéro).
- L'alternative désignée par `+` est moins prioritaire que la superposition. Ainsi, la chaîne `AB+C` désigne l'alternative entre un tas décrit par `AB` et un tas décrit par `C`.
- Les parenthèses sont autorisées. La chaîne `A(B+C)` désigne un tas dont la partie basse est décrite par `A` et la partie haute par `B+C`.

Exemples

1. `InitTas(1,'')` déclare le tas numéro 1 vide.
2. `InitTas(2,'TCK')` décrit le tas numéro 2 contenant de bas en haut un ♣, un ♡ et un ♢.
3. `InitTas(3,'T+P')` décrit le tas numéro 3 contenant une carte qui est soit un ♣ soit un ♠.
4. `InitTas(4,'[T+P]')` décrit le tas numéro 4 contenant un nombre quelconque, non déterminé, de cartes de couleur ♣ ou ♠.
5. `InitTas(1,'T+[P]CC')` décrit le tas numéro 1 contenant soit une seule carte de couleur ♣, soit un nombre quelconque de ♠ surmonté de deux ♡.

Action

La seule action permettant de modifier l'état des tas est le déplacement de la carte située au sommet d'un tas vers le sommet d'un autre tas. Cette action est ordonnée par un appel à la procédure à deux paramètres :

`DeplacerSommet(n,p)`

où `n` est le numéro du tas duquel on prend une carte, et `p` est celui du tas sur lequel on la pose.

Exemple :
`DeplacerSommet(2,4)` a pour effet de déplacer la carte au sommet du tas 2 pour la poser au sommet du tas 4.

Contrainte d'utilisation : il n'est pas permis de déplacer une carte située sur un tas vide. Toute tentative d'action `DeplacerSommet(n,p)` déclenche l'exception `Tas_Vide` si le tas `n` est vide.

Tests sur les tas

Certains traitements nécessitent des tests. Pour cela on dispose de fonctions.

Test de vacuité

Pour tester si un tas est vide, on fera appel à la fonction

`TasVide(n)`

qui donne la valeur `vrai` (`true` en anglais, et en PASCAL) si le tas numéro `n` est vide, `faux` (`false`) dans le cas contraire.

On peut faire le test contraire en faisant appel à la fonction

`TasNonVide(n)`

Test sur la couleur

Les quatre couleurs sont décrites dans l'unité `cartes` par quatre valeurs désignées par `TREFLE`, `PIQUE`, `COEUR` et `CARREAU`.

Il est possible de connaître la couleur au sommet d'un tas en faisant appel à la fonction

`CouleurSommet(n)`

qui donne la couleur de la carte située au sommet du tas numéro `n`.

Exemple :

```
if CouleurSommet(2)=PIQUE then ...
```

Contrainte d'utilisation : il est normalement dénué de sens de tester la couleur de la carte située au sommet d'un tas vide. Toute tentative d'appel à `CouleurSommet(n)` déclenche l'exception `Tas\_Vide` si le tas `n` est vide.

Quatre autres fonctions permettent aussi de tester la couleur du sommet d'un tas

`SommetTrefle(n)`,`SommetCarreau(n)`
`SommetCoeur(n)`,`SommetPique(n)`

qui donnent la valeur `vrai` si le sommet du tas `n` est un ♣ (♢, ♡, ♠).

Elles sont soumises à la même contrainte d'utilisation que la fonction `CouleurSommet`, et déclenchent la même exception en cas de non respect de cette contrainte.

Comparaison des valeurs

Une fonction permet de comparer la valeur des cartes au sommet de deux tas

`Superieur(n,p)`

qui donne la valeur `vrai` si la carte au sommet du tas numéro `n` a une valeur supérieure ou égale à celle du tas numéro `p`, et la valeur `faux` dans le cas contraire.

Contrainte d'utilisation : on ne peut comparer les cartes situées au sommet de deux tas que s'ils ne sont pas vides. Toute tentative d'appel à `Superieur(n,p)` déclenche l'exception `Tas\_Vide` si l'un des tas `n` ou `p` est vide.

Contrôle de l'affichage

Mode d'affichage

Lors de l'exécution du programme, chaque action est affichée, suivie du nouvel état des tas en résultant.

```
InitTas (Le_Tas => 1, l_expression => 'TT')
+----------+----------+----------+----------+
|   dame T |          |          |          |
|    5   T |          |          |          |
|  Tas 1   |  Tas 2   |  Tas 3   |  Tas 4   |
+----------+----------+----------+----------+
```

```
DeplacerSommet (TasDepart => 1, TasArriv => 2)
+----------+----------+----------+----------+
|    5   T |   dame T |          |          |
|  Tas 1   |  Tas 2   |  Tas 3   |  Tas 4   |
+----------+----------+----------+----------+

DeplacerSommet (TasDepart => 1, TasArriv => 2)
+----------+----------+----------+----------+
|          |    5   T |          |          |
|          |   dame T |          |          |
|  Tas 1   |  Tas 2   |  Tas 3   |  Tas 4   |
+----------+----------+----------+----------+
```

Il est ainsi possible de rediriger l'affichage vers un fichier texte, et d'en conserver ainsi une trace.

```
$ ./Exo1 > Exo1.result
```

Vitesse d'exécution

On peut modifier la vitesse d'exécution des déplacements par l'instruction

`lenteur := n`

Par défaut `lenteur` vaut 500. Plus le nombre `n` est grand, plus le programme s'exécute lentement.

Exemple :
`lenteur := 250` accélère le programme.

Un exemple de programme

Toutes les fonctions et procédures du langage de manipulation de cartes se trouvent dans l'unité `cartes`. Afin de pouvoir faire appel à elles, il faut donc déclarer l'utilisation de cette unité en début de programme : **uses** `cartes`.

Le programme qui suit est un exemple de programme de déplacement vers le tas 2 des deux cartes se trouvant initialement sur le tas 1.

Listing 1 – Un exemple de programme

```
// auteur : EW
// date : 10 sept 2004
// objet : exemple de manipulation cartes
// Situation initiale
//   Tas 1 : 'TK', Tas 2 : '',
//   Tas 3 : '', Tas 4 : ''
// Situation finale
//   Tas 1 : '', Tas 2 : 'KT',
//   Tas 3 : '', Tas 4 : ''
program Exemple;

uses
  cartes;

begin
  // Initialisation des tas
  initTas(1,'TK');
  initTas(2,'');
  initTas(3,'');
  initTas(4,'');


  // déplacement des cartes
  deplacerSommet(1,2);
  deplacerSommet(1,2);

end.
```

Exercices

Ils sont classés en facile (☺), moyen (😐)et difficile (☹)(les premiers sont très faciles).

Chaque exercice décrit une situation initiale des quatre tas de cartes (dans la syntaxe de l'unité `cartes`), et la situation finale à atteindre.

Séquence

Exercice 1. (☺)

Situation initiale :
Tas 1 : `'TT'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : ” Tas 2 : `'TT'`
Tas 3 : ” Tas 4 : ”

Exercice 2. (☺)

Situation initiale :
Tas 1 : `'TK'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'KT'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Exercice 3. (☺)

Situation initiale :
Tas 1 : `'TKTK'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'KKTT'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Exercice 4. (☺)

Situation initiale :
Tas 1 : `'TKCP'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'PCKT'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Conditionnelle

Exercice 5. (☺)

Situation initiale :
Tas 1 : `'T+P'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : ” Tas 2 : `'[T]'`
Tas 3 : `'[P]'` Tas 4 : ”

Exercice 6. (☺)

Situation initiale :
Tas 1 : `'(T+K+C+P)(T+K+C+P)'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : ” Tas 2 : ”
Tas 3 : `'(T+K+C+P)(T+K+C+P)'`↑ Tas 4 : ”

le symbole ↑ signifiant que les cartes sont dans l'ordre croissant (i.e. la carte du dessous a une valeur inférieure ou égale à celle du dessus).

Exercice 7. (☺)

Situation initiale :
Tas 1 : `'T+K+C+P'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[C]'` Tas 4 : `'[P]'`

Exercice 8. (☺)

Situation initiale :
Tas 1 : `'(T+K+C+P)(T+K+C+P)'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'[K+C]'` Tas 2 : `'[T+P]'`
Tas 3 : ” Tas 4 : ”

Itération

Exercice 9. (☺)

Situation initiale :
Tas 1 : `'[T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : ” Tas 2 : `'[T]'`
Tas 3 : ” Tas 4 : ”

Exercice 10. (😐)

Situation initiale :
Tas 1 : `'[K+C][T+P]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'[T+P][K+C]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Exercice 11. (😐)

Situation initiale :
Tas 1 : `'[K]'` Tas 2 : `'[T]'`
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : `'[K]'` Tas 2 : ”
Tas 3 : `'[KT]'` Tas 4 : ”
ou bien :
Tas 1 : ” Tas 2 : `'[T]'`
Tas 3 : `'[KT]'` Tas 4 : ”

Exercice 12. (😐)

Situation initiale :
Tas 1 : `'[T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”
Situation finale :
Tas 1 : ” Tas 2 : `'[T]'`
Tas 3 : `'[T]'` Tas 4 : ”

le nombre de cartes des deux tas 2 et 3 différant d'au plus 1 dans la situation finale.

Exercice 13. (☺☺)

Situation initiale :

Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[P]'` Tas 4 : ”

Situation finale :

Tas 1 : ” Tas 2 : `'[T]'`
Tas 3 : `'[K]'` Tas 4 : `'[P]'`

En faire deux versions, la seconde utilisant une procédure `vider_tas(depart,arrivee)` qui vide le tas `depart` sur le tas `arrivee`.

Exercice 14. (☺_)

Situation initiale :

Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[C]'` Tas 4 : `'[P]'`

Situation finale :

Tas 1 : `'[P]'` Tas 2 : `'[T]'`
Tas 3 : `'[K]'` Tas 4 : `'[C]'`

Exercice 15. (☺_)

Situation initiale :

Tas 1 : `'[T][K][C][P]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : `'[P][C][K][T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Exercice 16. (☺_)

Situation initiale :

Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[P]'` Tas 4 : ”

Situation finale :

Tas 1 : ” Tas 2 : ”
Tas 3 : ” Tas 4 : `'[TKP][`XY`][`Z`]'`

où X et Y désignent les deux couleurs restantes lorsque l'une des couleurs manque, et Z désigne la couleur restante lorsque X ou Y manque.

Exercice 17. (☺_)

Situation initiale :

Tas 1 : `'[T+K+C+P]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[C]'` Tas 4 : `'[P]'`

Exercice 18. (☺_)

Situation initiale :

Tas 1 : `'T[T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : ” Tas 2 : `'T'`$-$
Tas 3 : `'[T]'` Tas 4 : ”

le symbole $-$ indique que la carte est de valeur minimale.

Exercice 19. (☺☹)

Situation initiale :

Tas 1 : `'[T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : ” Tas 2 : `'[T]'`$\uparrow$
Tas 3 : ” Tas 4 : ”

le symbole $\uparrow$ signifiant que les cartes sont rangées par ordre croissant de valeurs de bas en haut.

Exercice 20. (☺_)

Situation initiale :

Tas 1 : `'[T+K]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : `'[`X`][`Y`]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Le symbole X désigne la couleur (♣ ou ♢) la plus nombreuse, l'autre couleur étant désignée par Y.

Exercice 21. (☺☹)

Situation initiale :

Tas 1 : `'K[T]'` Tas 2 : ”
Tas 3 : ” Tas 4 : ”

Situation finale :

Tas 1 : `'[T+K]'` Tas 2 : `'[T+K]'`
Tas 3 : `'[T+K]'` Tas 4 : `'[T+K]'`

les trèfles étant équitablement répartis sur les quatre tas, l'unique carreau se trouvant n'importe où.

Remarque : ce problème est infaisable sans le carreau.

Exercice 22. (☺☹)

Situation initiale :

Tas 1 : `'[T]'` Tas 2 : `'[K]'`
Tas 3 : `'[C]'` Tas 4 : `'[P]'`

Situation finale :

Tas 1 : `'[T]`$\uparrow$`'` Tas 2 : `'[K]`$\uparrow$`'`
Tas 3 : `'[C]`$\uparrow$`'` Tas 4 : `'[P]`$\uparrow$`'`

le symbole $\uparrow$ signifiant que les cartes sont rangées par ordre croissant de valeurs de bas en haut.