Procédures

1 Les procédures

Une *procédure* permet de créer une nouvelle action, ou encore d'*abstraire* et *nommer* une suite d'instructions. Une fois définie, une procédure peut être utilisée comme une nouvelle *instruction* du langage.

Comme toute instruction, un appel à une procédure modifie l'état courant de l'environnement : c'est un *effet de bord*.

1.1 Spécification d'une procédure

Spécifier une procédure, c'est
- choisir un identificateur pour la nommer,
- préciser le nombre de paramètres, leur mode de passage[1], leur type, et les nommer,
- indiquer le rôle joué par ces paramètres,
- indiquer les conditions d'utilisation (CU) que doivent vérifier les paramètres lors d'un appel à la procédure,
- et indiquer l'effet de la procédure sur l'environnement (effet de bord).

1.2 Déclaration d'une procédure en Pascal

```
procedure <Nom_Procedure>(<liste_parametres>);
  (* déclarations *)
begin
  (* séquence d'instructions *)
end {proc}
```

où
- <Nom_Procedure> est un identificateur,
- <liste_parametres> est la liste des *paramètres formels*.

L'*entête* de la procédure est la première ligne de sa déclaration.

Où ? La déclaration d'une procédure se fait dans la partie déclaration d'un programme (ou d'une procédure).

Exemples

```
// rassemble toutes les cartes sur le tas 1
procedure ToutMettreSurTas1;
begin
  // vider le tas 2 sur le tas 1
  while TasNonVide(2) do
  begin
    DeplacerSommet(2,1);
  end {while};

  // vider le tas 3 sur le tas 1
  while TasNonVide(3) do
  begin
    DeplacerSommet(3,1);
  end {while};
```

[1] les différents modes de passage (`in`, `out`, `var`) seront étudiés ultérieurement

```
  // vider le tas 4 sur le tas 1
  while TasNonVide(4) do
  begin
    DeplacerSommet(4,1);
  end {while};

end {TotMettreSurTas1}
```

Cette procédure a pour nom `ToutMettreSurTas1`, et n'a aucun paramètre.

Les deux premières lignes (le commentaire et l'entête) établissent la spécification de la procédure :

- l'entête donne le nom de la procédure (il est destiné à la fois au programmeur et au compilateur),
- le commentaire indique l'effet de la procédure sur son environnement (il est destiné uniquement au programmeur, le compilateur ignorant les commentaires).

```
// procédure pour vider les cartes du tas numéro depart
// sur le tas numéro arrivee
// CU : TasVide(depart) ou non(depart=arrivee)
//      le tas depart doit être différent du tas arrivee
//      ou bien le tas depart doit être vide
procedure ViderTas(depart : TasPossibles ; arrivee : TasPossibles);
begin
  while TasNonVide(depart) do
  begin
    DeplacerSommet(depart,arrivee);
  end {while};
end {ViderTas}
```

Les deux premières lignes (le commentaire et l'entête) établissent la spécification de la procédure :

- l'entête donne le nom de la procédure, ainsi que le nom et le type des paramètres (il est destiné à la fois au programmeur et au compilateur),
- le commentaire indique l'effet de la procédure sur son environnement, et le rôle joué par les paramètres (il est destiné uniquement au programmeur, le compilateur ignorant les commentaires).

Cette procédure a pour nom `ViderTas` et possède deux paramètres (`depart` et `arrivee`) qualifiés de *formels*, car lors de la conception (ou écriture) de cette procédure, ces paramètres n'ont aucune valeur. Ils servent à *nommer* les (futures) valeurs qui seront passées lors d'un appel à la procédure.

Les paramètres formels sont définis

- à l'aide d'un *identificateur* ou *nom*, (ici `depart` et `arrivee`),
- d'un mode (**in**, **out**, **var** que l'on étudiera plus tard),
- et d'un type (ici `TasPossibles`).

Lorsque plusieurs paramètres formels ont même type et sont de même mode, on peut rassembler leurs identificateurs avant le type. Par exemple,

```
procedure ViderTas(depart,arrivee : TasPossibles);
```

1.3 Appel à une procédure

Un appel à une procédure est une instruction. Le résultat de son exécution est une modification de l'état courant de l'environnement.

Pour faire appel à une procédure sans paramètre, il suffit d'écrire son nom. Par exemple :

```
ToutMettreSurTas1 ;
```

Cette instruction a pour effet de mettre toutes les cartes sur le tas 1.

Pour faire appel à une procédure avec paramètres, il suffit d'écrire son nom accompagné des *paramètres effectifs*. Exemple :

```
ViderTas(1,2) ;
```

Cette instruction vide le tas 1 sur le tas 2. Le numéros 1 et 2 sont appelés paramètres *effectifs*

Les paramètres effectifs d'un appel à une procédure doivent satisfaire les conditions d'utilisation.

1.4 Intérêt des procédures

- reflet de l'analyse du problème
- modularité
- lisibilité des programmes
- factorisation du code

1.5 Méthodologie

SPECIFIER, SPECIFIER, SPECIFIER

2 Exercices

Exercice 1. Écrire l'entête de la procédure `deplacerSommet` de l'unité cartes. (Le type qui définit les couleurs se nomme `Couleurs`, et celui qui définit les tas se nomme `TasPossibles`)

Exercice 2. Expliquez la condition d'utilisation de la procédure `ViderTas`.

Exercice 3.

Question 1. Reprendre la procédure `ToutMettreSurTas1` en utilisant la procédure `ViderTas`. (attention à l'ordre des déclarations de ces procédures)
Question 2. Faire une version paramétrée par le numéro du tas sur lequel on veut tout mettre.

Exercice 4. Les exercices de manipulation de cartes.