Structures de données

Exercice sur la simulation des pointeurs

Certains langages de programmation ne permettent pas la manipulation de zones de mémoires repérées par des pointeurs et allouées dynamiquement (par exemple, Fortran ou Cobol). En général, ils offrent cependant la possibilité de manipuler des tableaux. Cet exercice a pour but d'étudier l'implantation de structures dynamiques dans des tableaux. Considérons les déclarations suivantes:

```
Zones : array (1..1000) of record
    Suivant : Natural;  -- indice de la zone libre suivante
    Contenu : Element;
end record;
Premiere_Zone : Natural;
procedure Initialiser_Zone;
function Allouer_Zone return Natural;
 -- retourne un numéro de zone libre s'il y en a, et 0 sinon
procedure Restituer_Zone (P: Natural);
 -- restitue la zone de numéro P qui redevient libre
```

A- Quelle signification doit on donner à un numéro de zone égal à 0? Cela est-il cohérent avec les déclarations?

B- Comment se ferait l'accès aux données conservées dans une zone de numéro donné?

C- En expliquant votre raisonnement, proposer le corps de l'opération `Initialiser_Zone`.

D- Proposer le corps de l'opération `Allouer_Zone`.

E- Proposer le corps de l'opération `Restituer_Zone`.

Corrigé

Question A

Il ressort de la déclaration du tableau `Zones`, que les numéros de zone valides, i.e. les indices du tableau, sont compris entre 1 et 1000. Toute autre valeur n'est pas un numéro de zone. La fonction `Allouer_Zone` retourne 0 dans le cas où aucune zone n'est libre, c'est-à-dire précisément lorsqu'elle ne peut retourner un numéro de zone acceptable. Or dans l'utilisation des pointeurs, il est nécessaire de disposer d'une valeur particulière qui ne repère aucune zone, et que l'on désigne souvent par `null`. Dans ce cas-ci, 0 peut remplir ce rôle, et cette interprétation est cohérente avec la déclaration de la fonction `Allouer_Zone`.

Question B

Si une donnée est conservée dans une zone de numéro `i`, qui doit évidemment être compris entre 1 et 1000, cela veut dire que l'élément du tableau d'indice `i` du tableau `Zones` contient cette donnée. On y accède donc par: `Zones(i).Contenu`.

Question C

A l'initialisation, toutes les zones sont libres. Par la suite, il faudra savoir celles qui sont libre et celles qui ne le sont pas. Evidemment, on peut penser que l'allocation des zones se fera par numéro croissant, jusqu'à la dernière, les zones libres se suivant toutes. Cependant lorsqu'il y aura restitution d'une zone, ce peut être n'importe laquelle parmi celles qui sont occupées, et il n'y a aucune raison qu'elle précède ou suive celles qui étaient déjà libres. Le regroupement des zones libres séquentiellement ne sera donc plus possible.

On peut imaginer que chaque zone libre repère la zone libre suivante, comme le laisse penser le commentaire associé à la déclaration du champ `Suivant`. A l'état initial, il faut alors organiser l'ensemble des zones libres pour que chacune sauf la dernière repère sa suivante.

```
procedure Initialiser_Zone is
begin
  for I in 1 .. 999 loop
    Zones(I).Suivant := I + 1;
  end loop;
  Zones(1000).Suivant := 0; -- null: la dernière ne repère rien
  Premiere_Zone := 1; -- la première zone libre
end Initialiser_Zone;
```

Question D

On alloue la première zone libre éventuelle, et on détermine la suivante.

```
function Allouer_Zone return Natural is
  Numero_alloue : Natural := Premiere_Zone; -- le numéro de la
              -- première zone libre si elle existe, et 0 sinon
begin
  if Premiere_Zone /= 0 then -- la zone existe
    Premiere_Zone := Zones(Premiere_Zone).Suivant;
      -- On détermine la nouvelle zone libre
  end if;
  return Numero_alloue;
end Allouer_Zone;
```

Question E

Lorsqu'une zone redevient libre, il s'agit de la remettre dans la structure de façon à permettre sa réattribution ultérieure si nécessaire. Le plus simple est de décider qu'elle est maintenant la première zone libre et que l'ancienne première zone libre devienne la seconde.

```
procedure Restituer_Zone (P: Natural);
begin
  Zones(P).Suivant := Premiere_Zone;
                  -- L'ancienne première libre devient la seconde
  Premiere_Zone := P;          -- P devient la nouvelle première
end Restituer _Zone;
```