Valeur d'Accueil et de Reconversion en Informatique

Corrigé du partiel de février 2004

Durée 2 heures
Tout document écrit autorisé

Conseil : consacrer 1 heure 30 à la partie programmation et 30 minutes à la partie architecture et systèmes

I – Partie Programmation (15 points)

Exercice 1 : Exploration minière (3.5 points)

On dispose de données obtenues par lecture infrarouge, d'une portion de désert où la présence d'un gisement d'or est suspectée. Ces données devraient permettre de délimiter un éventuel gisement aurifère.
Chaque portion du désert est représentée par une matrice 8*8. A chaque élément de la matrice correspond une donnée. Une donnée supérieure à la moyenne de ses points limitrophes (situés aux 4 points cardinaux) indique la présence d'or.

Voici un exemple de relevé représenté sous la forme d'une matrice sur lequel est basé cette étude :

indices lignes/colonnes	1	2	3	4	5	6	7	8
1	21	21	22	30	40	21	34	45
2	21	**22**	**23**	**30**	**45**	**21**	**37**	40
3	22	**23**	**24**	**45**	**46**	**47**	**38**	39
4	22	**23**	**24**	**35**	**46**	**47**	**38**	38
5	23	**24**	**25**	**36**	**46**	**49**	**37**	36
6	23	**24**	**25**	**37**	**39**	**48**	**36**	35
7	23	**24**	**25**	**25**	**26**	**25**	**26**	25
8	23	25	26	27	28	29	30	31

On suppose que la redondance est suffisante pour ne pas se préoccuper des points situés au bord de la grille. Seules les mesures indiquées en gras sont concernées par le calcul.

Le programme est chargé de créer la carte du gisement sous la forme d'une matrice représentant la région explorée dans laquelle les données supérieures à la moyenne seront indiquées par le caractère `'*'` et les autres points par le caractère `'-'`.

Question 1 (0,5 points)

Déclarer le type `Releve` permettant la représentation de tous les relevés de mesures.

Solution

```
subtype Indice is Integer range 1..8;
type Releve is array (Indice,Indice) of Natural;
```

Question 2 (0,5 points)

Déclarer le type `Gisement` permettant la représentation des cartes de gisement aurifère.

Solution

```
type Gisement is array (Indice,Indice) of Character;
```

Question 3 (0.5 points)

Déclarer la fonction calcul qui étant donné un relevé de mesure crée la carte du gisement correspondante.

Solution

```
function calcul( r : Releve ) return Gisement
```

Question 4 (2 points)

Déclarer le corps de cette fonction.

Solution

```
function calcul( r : Releve ) return Gisement is
somme:Positive;
c:Gisement;
begin
 for i in Indice'first+1..Indice'last-1 loop
   for j in Indice'first+1..Indice'last-1 loop
     -- somme des 4 points cardinaux :
     somme:=r(i-1,j)+r(i+1,j)+r(i,j-1)+r(i,j+1);
     if (r(i,j)>somme/4)
     then
        c(i,j):='*';
     else
        c(i,j):='-';
     end if;
   end loop;
 end loop;
 return c;
end calcul;
```

Exercice 2 : jeu des sept erreurs (3.5 points)

Le programme suivant comporte 7 erreurs de compilation.
Décrivez les erreurs en donnant le numéro de la **ligne** où l'erreur est **détectée** et la **cause** de chacune d'entre elles.

```
1     with Ada.Float_Text_io;

2     procedure sept_erreurs is
3
4       type Arbre is access Cellule;
5       type cellule is
6         record
7           valeur:Float;
8           gauche,droite:Arbre;
9         end record;
10      type Vecteur is array( Positive range <> ) of Float;

11      procedure inserer( x: in Float; a: in Arbre ) is
12      begin
13        if a=null
14        then
15           a:=new Cellule'( x,null,null );
16        else
17          if x<a.valeur
18          then
19             inserer( x,a.gauche );
20          else
21             inserer( x,a.droite );
22          end if;
23      end inserer;
24
25      procedure put( a: in Arbre ) is
26      begin
27         if a/=null
28         then
29            put( a.gauche );
30            put( a.valeur,2,1,0 );
31            put( a.droite );
32         end if;
33      end put;

34      v:Vecteur:=(3.5,7.8,9,2.2,6.3,0.7,7.6);
35      a:Arbre;

36    begin

37      for i in v loop
38        inserer(v(i),arbre);
39      end loop;
40      put(a);
41    end sept_erreurs;
```

Solution

ligne 4 : le type Cellule n'appartient pas à l'environnement. Il doit être déclaré.
ligne 11 : le paramètre a est passé en in. Il ne peut donc pas être affecté
ligne 23 : il manque un end if
ligne 30 : la procédure put sur des flottants n'appartient pas à l'environnement. Il faut soit
préfixer par Ada.Float_Text_io, soit insérer un use Ada.Float_Text_io;.
ligne 34 : erreur de typage. Certaines valeurs du vecteur sont des littéraux de type Integer
ligne 37 : v n'est pas un intervalle de valeur entière.
ligne 38 : le paramètre arbre est un type et non une variable

Exercice 3 : le programme à trous (8 points)

Le programme suivant est constitué de deux packages `elements` et `piles` et d'un programme de tests `client_piles`.

```
-- fichier elements.ads
package elements is
   type Element is new Integer;
end elements;

-- fichier piles.ads
with elements;use elements;
package piles is
   type Pile is limited private;
   pileVide:exception;
-- l'élément x est place au sommet de la pile c
1  procedure empiler( p: ?? Pile;x:?? Element );
-- l'élément placé au somment de la pile c est affecté à x
-- et enlevé de la pile
2  procedure depiler( p: ?? Pile;x:?? Element );
   function sommet( p: Pile ) return Element;
   function estVide( p : Pile ) return Boolean;
private
   type Cellule;
   type Pile is access Cellule;
   type Cellule is
      record
         valeur:Element;
         suivant:Pile;
      end record;
end piles;

-- fichier piles.adb
with Ada.Text_io;use Ada.Text_io;
package body piles is
   procedure empiler( p: ?? Pile;x:?? Element ) is
   begin
-- x est ajouté en tête de la pile p
3     ??????????????????????????????
   end empiler;
-- x reçoit la l'élément placé au sommet de la pile et est enlevé de la
pile
-- sauf si la pile est vide
-- dans ce cas, le message "opération impossible est affiché"
4  procedure depiler( p: ?? Pile;x:?? Element ) is
   begin
5      ????????????????????????????????????????
   end depiler;
-- la fonction sommet retourne l'élément situé au sommet de la pile.
-- la pile reste identique
   function sommet( p: Pile ) return Element is
   begin
6      ????????????????????????????????????????
   end sommet;
-- la fonction sommet retourne la valeur true si la pile est vide
-- false sinon
   function estVide( p : Pile ) return Boolean is
   begin
```

```
7      ??????????????????????????????????????
   end estVide;
end piles;

-- fichier client_piles.adb
with piles;use piles;
with elements;use elements;
with Ada.Text_io;use Ada.Text_io;
procedure client_piles is
   p:Pile;
   x:Element;
begin
   for i in 1..10 loop
-- empiler l'élément x dans p
8      ??????????????????????????????????????
   end loop;
   loop
   exit when estVide( p );
      depiler( p,x );
      put(Element'image( x ));
      put(' ');
   end loop;
end client_piles;
```

Question 1 (4 points)

Ce programme est incomplet, remplacez les lignes marquées par des ?? précédées du commentaire expliquant leur rôle.

Solution

```
-- fichier piles.ads
```

```
with elements;use elements;
package piles is
   type Pile is limited private;
   pileVide:exception;
   procedure empiler( p: in out Pile;x:in Element );
   procedure depiler(p: in out Pile;x:out Element);
   function sommet(p: Pile) return Element;
   function estVide(p : Pile) return Boolean;
private
   type Cellule;
   type Pile is access Cellule;
   type Cellule is
      record
        valeur:Element;
         suivant:Pile;
      end record;
end piles;

-- fichier piles.adb
with Ada.Text_io;use Ada.Text_io;
package body piles is
   procedure empiler(p: in out Pile;x:in Element) is
   begin
      p:=new Cellule'(x,p);
   end empiler;
   procedure depiler(p: in out Pile;x:out Element) is
   begin
      if p/=null
      then
         x:=p.valeur;
         p:=p.suivant;
      else
         raise pileVide;
      end if;
   exception
      when pileVide=>put_line("opération impossible");
   end depiler;
   function sommet(p: Pile) return Element is
   begin
      return p.valeur;
   end sommet;
   function estVide(p : Pile) return Boolean is
   begin
      return p=null;
   end estVide;
end piles;

-- fichier client_piles.adb
with piles_tab;use piles_tab;
with elements;use elements;
with Ada.Text_io;use Ada.Text_io;
procedure client_piles is
   p:Pile;
   x:Element;
begin
   for i in 1..10 loop
      empiler(p,Element(i));
   end loop;
   loop
   exit when estVide(p);
      depiler(p,x);
```

```
      put(Element'image(x));
      put(' ');
   end loop;
end client_piles;
```

Question 2 (4 points)

Modifier ce programme pour prendre en compte une nouvelle représentation de pile. Le type `Pile` est maintenant définie comme un tableau d'éléments dont la taille est fixée par la constante `MAX=100` . On ne se préoccupera pas du cas où une pile devient pleine.

Solution

Le fichier `client_piles.adb` n'est pas modifié
Seule la partie privée de `piles.ads` est modifiée

```
-- fichier piles.ads
with elements; use elements;
package piles is
   type Pile is limited private;
   pileVide : exception;
   procedure empiler( p: in out Pile; x: in Element );
   procedure depiler( p: in out Pile; x: out Element );
   function sommet( p: Pile ) return Element;
   function estVide( p : Pile ) return Boolean;
private
   MAX:constant Integer:=100;
   subtype Indice is Integer range 0..MAX;
   type TabPile is array( Indice ) of Element;
   type Pile is
      record
            indiceCourant : Indice:=0;
            tableau : TabPile;
      end record;
end piles;

-- fichier piles.adb
with Ada.Text_io;use Ada.Text_io;
with elements;use elements;

package body piles_tab is
   procedure empiler( p: in out Pile; x: in Element ) is
   begin
      p.indiceCourant:=p.indiceCourant+1;
      p.tableau(p.indiceCourant):=x;
   end empiler;
   procedure depiler( p: in out Pile; x: out Element ) is
   begin
      if p.indiceCourant/=0
      then
         x:=p.tableau( p.indiceCourant );
         p.indiceCourant:=p.indiceCourant-1;
      else
         raise pileVide;
      end if;
```

```
   exception
      when pileVide=>put_line( "opération impossible" );
   end depiler;
   function sommet( p: Pile ) return Element is
   begin
      return p.tableau( p.indiceCourant );
   end sommet;
   function estVide( p: Pile ) return Boolean is
   begin
      return p.indiceCourant=0;
   end estVide;
end piles_tab;
```

II – Partie Architecture (2.5 points)

Question 1 (0,5 point)

Quelle est la définition d'un ordinateur ?

a) un calculateur électronique
b) un calculateur automatique à programme enregistré
c) un automate contrôlé par programme
d) un automate susceptible de prendre des décisions

Réponse b)

Question 2 (0,5 point)

Qu'est-ce qu'une architecture en pipeline ?

a) une organisation du processeur permettant que des instructions successives d'un même programme soient traitées en même temps par des sous-unités indépendantes.
b) une organisation basée sur une mémoire cache, que l'on intercale entre le processeur et la mémoire centrale.
c) des couples processeur+mémoire indépendants et inter-connectés

Réponse a)

Question 3 (0,5 point)

Quelle valeur choisir pour définir le temps de cycle élémentaire d'un processeur ?

a) c'est la valeur imposée par l'horloge du processeur.
b) c'est la micro-opération la plus lente qui doit définir le temps de cycle.

Réponse b)

Question 4 (0,5 point)

Quelle est la plus grosse mémoire ? (0,5 point)

a) une mémoire adressable sur 16 bits, délivrant des mots de 7 bits
b) une mémoire adressable sur 15 bits, délivrant des mots de 15 bits
c) une mémoire de 55 Ko

Réponse c)

Question 5 (0,5 point)

A la réception d'une interruption matérielle externe, que doit faire le processeur ?

a) sauvegarder le contenu du compteur ordinal (PC) et de tous ses autres registres dans une zone de mémoire spécifique, puis charger dans le PC l'adresse de début du sous-programme d'interruption
b) envoyer un signal d'autorisation au périphérique qui a généré l'interruption pour que ce dernier puisse transférer ses informations en mémoire centrale. Attendre ensuite que ce transfert se termine pour reprendre l'exécution du programme en cours.

Réponse a)