S2 maths

STL1AP2005

Examen de mai 2005

durée 2h - documents non autorisés

Exercice 1 : *Compréhension du langage* (1/2H)

Q 1 . On suppose dans cette question déclarées les procédures, fonctions et variables suivantes :

```
procedure p(const x : CARDINAL; out y : BOOLEAN);
function f(const x : CARDINAL) : CARDINAL;

var
  x,y : BOOLEAN;
  z,t : CARDINAL;
```

Dans ce contexte, quelles sont les instructions valides parmi celles ci-dessous ?

1. p(z,x);
2. p(1,true);
3. p(2*z,x);
4. p(z,x **and** y);
5. a := p(z,x);
6. f(z) := t;
7. z := f(t);
8. z := f(2*z);
9. **if** f(z) **then** ...
10. f(z);
11. p(f(z),x);
12. z := f(p(z,x));

Q 2 . On suppose dans cette question déclarées les procédures, fonctions et variables suivantes :

```
procedure p(const x : INTEGER; var y : INTEGER);
var
  z : INTEGER;
begin
   z := x+y;
   y := z-2*y;
end {p};

var
  x,y,z : INTEGER;
```

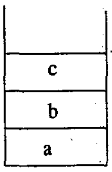
En supposant les variables initialisées par $\{x=1, y=2, z=3\}$, indiquez la valeur de ces variables après chacune des instructions suivantes :

1. p(x,y);
2. p(y,z);
3. p(x+y,z);

Exercice 2 : (20MN)

Q 1 . Rappelez les spécifications des opérations primitives sur les piles.

Q 2 . Expliquez comment on peut réaliser une implémentation des piles avec des listes. Indiquez en particulier comment représenter la pile ci-dessous avec une liste.



Q 3 . Déclarez le type T_PILE.

Q 4 . Réalisez les opérations empiler et depiler en utilisant les opérations sur les listes.

Exercice 3 : *Polynômes* (1H10)

Dans cet exercice on souhaite écrire quelques fonctions et procédures permettant de manipuler des polynômes P à coefficients réels de degré d au plus égal à 30.

$$P(x) = \sum_{i=0}^{d} c_i x^i$$

Pour cela, on va définir un type T_POLYNOME comme un couple de deux données :

1. un tableau de réels contenant les coefficients du polynôme, indexé par le type T_INDICE,
2. un entier précisant le degré du polynôme représenté (on conviendra que le polynôme nul a pour degré 0).

où le type T_INDICE est défini par

```
const DEGMAX = 30;
type  T_INDICE = 0..DEGMAX;
```

Q 1 . En vous inspirant de la fonction mult_scalaire qui permet d'afficher un polynôme à l'écran, déclarez le type T_POLYNOME.

```
// mult_scalaire(a,p) = ap
function mult_scalaire(a : REAL; p : T_POLYNOME) : T_POLYNOME;
var
  i : T_INDICE;
  q : T_POLYNOME;
begin
  if a=0 then begin
    q.degre := 0;
    q.coeff := 0;
  end else begin
    q.degre := p.degre;
    for i := 0 to p.degre do begin
      q.coeff[i] := a*p.coeff[i];
    end {for};
  end {if};
  mult_scalaire := q;
end {mult_scalaire};
```

Q 2 . En dessinant un schéma, donnez la représentation selon le type T_POLYNOME du polynôme $x^3 - x + 1$.

Q 3 . On veut maintenant réaliser une fonction qui calcule la valeur $P(a)$ d'un polynôme P en un réel a.

```
// evalue(a,p)=p(a)
function evalue(a: REAL; p : T_POLYNOME) : REAL;
```

Q 3.1. Donnez une première réalisation de cette fonction, qui effectue cette évaluation en calculant tous les termes de la forme $c_i a^i$ et les somme tous. Pour cela vous pourrez utiliser la fonction puissance.

```
// puissance(x,n) = x^n
function puissance(x : REAL; n : CARDINAL) : REAL;
var
begin
  p := 1;
  for i := 1 to n do begin
    p := p*x;
  end {for}
  puissance := p;
end {puissance};
```

Q 3.2. On peut évaluer différemment un polynôme en suivant un algorithme connu sous le nom de schéma de Hörner fondé sur l'égalité

$$c_d x^d + c_{d-1}x^{d-1} + \ldots + c_1 x + c_0 = (\ldots((c_d x + c_{d-1})x + c_{d-2})x + \ldots)x + c_0$$

Par exemple

$$5x^3 + 2x^2 - 3x + 1 = ((5x+2)x - 3)x + 1$$

Q 3.3. Déterminez en fonction de d les nombres d'additions et de multiplications de réels effectués dans l'évaluation d'un polynôme de degré d pour chacune des deux réalisations précédentes.

Q 4 . Réalisez la fonction calculant la somme de deux polynômes

```
// add(p1,p2)=p1+p2
function add(p1,p2 : T_POLYNOME) : T_POLYNOME;
```

Attention au degré! ($p_1 = x^3 - x + 1, \quad p_2 = -x^3 + x^2 - 1$)

Q 5 . Réalisez une procédure nommée deriver calculant le polynôme dérivé de celui passé en paramètre

```
{$p=x^3-x+1$}
deriver(p);
{$p=3x^2-1$}
```

Chercher une solution sans variable locale de type T_POLYNOME.