Département STPI Lundi 11 Avril 2011

I3 - Algorithmique

Durée : *1h30*
Documents autorisés : **AUCUN** (calculatrice comprise)

Remarques :
- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier.

1 Compréhension du cours (10 points)

La figure 1 présente l'analyse decendante (vue en cours) du problème de transformation d'une chaîne de caractères (représentant peut-être un réel positif de la forme *"partie_entière"* ou *"partie_entière,partie_décimale"* avec au moins un chiffre dans *partie_entière* et *partie_décimale*) en réel.

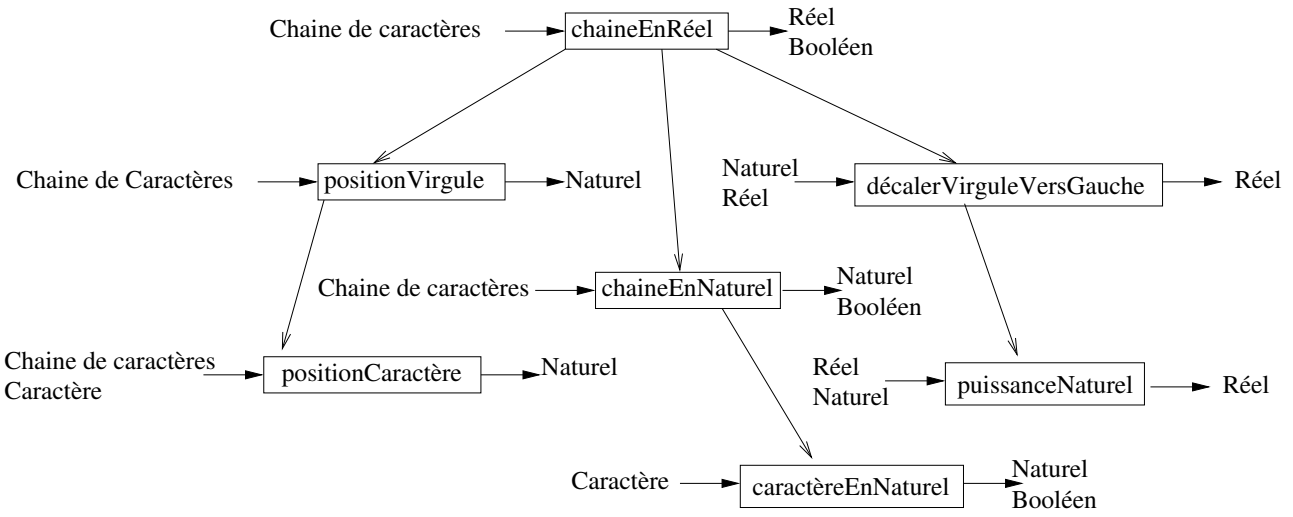

FIGURE 1 – Analyse descendante

tel que les (sous-)problèmes :

positionVirgule permet d'obtenir la position du caractère virgule (`','`) dans une chaîne ch. Il retourne 0 si la virgule n'est pas présente ;

chaineEnNaturel permet d'essayer de transformer une chaîne ch représentant un naturel en un naturel ;

décalerVirguleVersGauche permet de décaler la virgule d'un réel x de n rangs vers la gauche ;

positionCaractère permet d'obtenir la position d'un caractère c dans une chaîne ch. Il retourne 0 si le caractère n'est pas présent ;

caractèreEnNaturel permet d'essayer de transformer un caractère c en un naturel ;

puissanceNaturel permet d'élever un réel x à une puissance n naturel.

1.1 Conception préliminaire

Donnez la signature des fonctions et procédures correspondant aux (sous-)problèmes identifiés par cette analyse descendante.

Solution proposée :

- **procédure** chaineEnRéel (**E** c : **Chaine de caracteres**,**S** val : **Reel**, erreur : **Booleen)**
- **fonction** positionVirgule (c : **Chaine de caracteres) : Naturel**
- **fonction** positionCaractere (ch : **Chaine de caracteres**, c : **Caractere) : Naturel**
- **procédure** chaineEnEntier (**E** ch : **Chaine de caracteres**,**S** val : **Entier**, erreur : **Booleen)**
- **procédure** caractereEnEntier (**E** c : **Caractere**,**S** val : **Entier**, erreur : **Booleen)**
- **fonction** decalerVirguleVersGauche (val : **Reel**, nbChiffre : **Naturel) : Reel**
- **fonction** puissanceNaturel (val : **Reel**, exposant : **Naturel) : Reel**

1.2 Conception détaillée

On suppose que l'on possède les fonctions suivantes :
- **fonction** longueur (uneChaine : **Chaine de caracteres) : Naturel**
- **fonction** iemeCaractere (uneChaine : **Chaine de caracteres**, position : **Naturel) : Caractere**
- **fonction** sousChaine (uneChaine : **Chaine de caracteres**, debut,fin : **Naturel) : Caractere**

1. Donnez le corps la procédure/fonction du (sous-)problème `positionCaractere` qui retourne la première position d'un caractère c s'il est présent dans la chaîne ch (la position du premier caractère de la chaîne ch vaut 1) et 0 sinon.
2. Donnez le corps de la procédure/fonction du (sous-)problème `chaineEnRéel` qui permet de transformer une chaîne ch en réel en sachant s'il y a eu ou pas une erreur.

Solution proposée :

fonction positionCaractere (ch : **Chaine de caracteres**, c : **Caractere) : Naturel**

Déclaration i : **Naturel**
trouvé : **Booleen**

debut
i ← 1
trouvé ← FAUX
tant que i ≤ longueurChaine(ch) et non trouve **faire**
si iemeCaractere(ch,i)=c **alors**
trouvé ← VRAI
sinon
i ← i+1
finsi
fintantque
si trouvé **alors**
retourner i
sinon
retourner 0
finsi
fin

procédure chaineEnRéel (**E** c : **Chaine de caracteres**,**S** val : **Reel**, erreur : **Booleen)**

Déclaration pos,partieEntiere,partieDecimale : **Naturel**

debut
pos ← positionVirgule(c)
si pos≠0 **alors**
chaineEnEntier(sousChaine(c,1,pos-1),partieEntiere,erreur)
si non erreur **alors**
chaineEnEntier(sousChaine(c,pos+1,longueurChaine(c)),partieDecimale,erreur)
si non erreur **alors**
val ← partieEntier+decalerVirguleVersGauche(partieDecimale,longueurChaine(c)-pos)
finsi

```
        finsi
    sinon
        chaineEnEntier(c,partieEntiere,erreur)
        si non erreur alors
            val ← partieEntier
        finsi
    finsi
fin
```

2 Développement limité (6 points)

Lorsque x est proche de 0, $(1+x)^a$ peut être approximé à l'aide du développement limité suivant :

$$1 + \sum_{i=1}^{n} \frac{\prod_{j=1}^{i}(a-j+1)}{i!} x^i$$

Donnez le corps de la fonction suivante qui calcule une approximation de $(1+x)^a$ jusqu'au rang n :

– **fonction** unPlusXExpA (x,a : **Reel**, n : **Naturel) : Reel**

Votre algorithme devra respecter les deux conditions suivantes :

– sa complexité doit être en $O(n)$;

– vous utiliserez uniquement des variables locales et les paramètres formels donnés (vous n'utiliserez aucune autre fonction, pas d'analyse descendante).

Solution proposée :

```
fonction unPlusXExpA (x,a : Reel, n : Naturel) : Reel
    Déclaration  numerateur,resultat, xPuissanceI : Reel
                 factI,i : Naturel
debut
    numerateur ← 1.0
    factI ← 1
    xPuissanceI ← 1.0
    resultat ← 1.0
    pour i ←1 à n faire
        numerateur ← numerateur*(a-i+1)
        factI ← factI*i
        xPuissanceI ← xPuissanceI*x
        resultat ← resultat+numerateur*xPuissanceI/iFact
    finpour
    retourner resultat
fin
```

3 Erreurs dans un programme Pascal (4 points)

Soit le programme pascal suivant :

```
1  program chiffrementCesar;
2
3  function caractereAdmissible(unCaractere : Char) : Boolean;
4  begin
5      caractereAdmissible:= (unCaractere>='A' and unCaractere<='Z') or (unCaractere
           >='a' and unCaractere<='z') or (unCaractere=' ')
6  end;
7
8  function decalage(unCaractere :  Char) : Integer;
```

```
 9 begin
10    if (unCaractere>='A') and (unCaractere<='Z') then
11       decalage:=(ord(unCaractere)-ord('A'))
12    else
13       if (unCaractere>='a') and (unCaractere<='z') then
14          decalage:=26+(ord(unCaractere)-ord('a'))
15       else
16          decalage:=52
17 end;
18
19 function chiffrerPhrase(unePhrase : String; uneCle : Char) : Chaine;
20 var
21    resultat      : String;
22    leDecalage    :  Integer;
23 begin
24    resultat:='';
25    leDecalage:=decalage(uneCle);
26    for i:=1 to length(unePhrase) do
27       if caractereAdmissible(unePhrase[i]) then
28          resultat:=resultat+chiffrerCaractere(unePhrase[i],leDecalage);
29    chiffrerPhrase:=resultat
30 end;
31
32 function chiffrerCaractere(unCaractere : Char; unDecalage : Integer) : Char;
33 var
34    resultat : Char;
35    i        : Integer;
36 begin
37    resultat:=unCaractere;
38    for i:=1. to unDecalage do
39       case resultat of
40         'Z' : resultat:='a';
41         'z' : resultat:=' ';
42         ' ' : resultat:='A';
43       else resultat:=succ(resultat)
44       end;
45    chiffrerCaractere:=resultat
46 end;
47
48 function finDeProgramme(unePhrase : String) : Boolean;
49 begin
50    finDeProgramme:=(upcase(unePhrase)='FIN.')
51 end;
52
53 function obtenirPhrase() : String;
54 var
55    resultat : Integer;
56 begin
57    write('Message a chiffrer :');
58    readln(resultat);
59    obtenirPhrase:=resultat
60 end;
61
62 function obtenirCle : Char;
63 var
64    resultat :  Char;
65 begin
66    write('La cle : ');
67    repeat
68       readln(resultat);
69    until caractereAdmissible(resultat);
70    obtenirCle:=resultat
71 end;
72
73 procedure machineAChiffrer();
74 var
75    laCle     : Char;
76    laPhrase : String;
```

```
77 begin
78     laCle:=obtenirCle();
79     laPhrase:=obtenirPhrase();
80     while not finDeProgramme(laPhrase) do
81     begin
82         writeln(chiffrerPhrase(laPhrase,laCle));
83         laPhrase:=obtenirPhrase();
84     end
85 end;
86
87 begin
88     machineAChiffrer()
89 end;
```

Sa compilation donne les informations suivantes :

```
1  $ fpc chiffrementCesar.pas
2  Free Pascal Compiler version 2.4.0-2 [2010/03/06] for x86_64
3  Copyright (c) 1993-2009 by Florian Klaempfl
4  Target OS: Linux for x86-64
5  Compiling chiffrementCesar.pas
6  chiffrementCesar.pas(5,44) Error: Operation "and" not supported for types "Char"
       and "Char"
7  chiffrementCesar.pas(5,87) Error: Operation "and" not supported for types "Char"
       and "Char"
8  chiffrementCesar.pas(19,68) Error: Identifier not found "Chaine"
9  chiffrementCesar.pas(26,9) Error: Identifier not found "i"
10 chiffrementCesar.pas(27,41) Error: Identifier not found "i"
11 chiffrementCesar.pas(28,39) Error: Identifier not found "chiffrerCaractere"
12 chiffrementCesar.pas(28,51) Error: Identifier not found "i"
13 chiffrementCesar.pas(38,11) Error: Incompatible types: got "Single" expected "
       SmallInt"
14 chiffrementCesar.pas(59,19) Error: Incompatible types: got "SmallInt" expected "
       ShortString"
15 chiffrementCesar.pas(82,45) Error: Can't read or write variables of this type
16 chiffrementCesar.pas(89,4) Fatal: Syntax error, "." expected but ";" found
17 Fatal: Compilation aborted
18 $
```

Justifiez chacune des erreurs et proposez une correction.

Solution proposée :

Ligne 6 Manque des parenthèses. Le pascal essaye de faire un *and* entre deux caractères

Ligne 7 *idem*

Ligne 8 Le pascal ne connaît pas le type `Chaine`. Il faut mettre `String` à la place.

Ligne 9 Le pascal ne connaît pas l'identifiant de variable *i*, il faut le déclarer comme étant un *Integer*.

Ligne 10 *idem*

Ligne 11 Le pascal ne connait l'identifiant de fonction `chiffrerCaractere`, car la fonction est déclarée (et définie) après. Il faut au moins la déclarer avant cet appel.

Ligne 12 *idem* ligne 10

Ligne 13 Le début de la boucle est une constante de type `Real` (1.) ce qui n'est pas possible en pascal.

Ligne 14 La variable `resultat` est déclarée comme `Integer` et le type de retour de la fonction est `String` : l'affectation n'est pas possible. Il faut déclarer `resultat` comme étant un `String`.

Ligne 15 La procédure `writeln` ne peut pas prendre comme argument une valeur de type `Chaine` (même problème que la ligne 8)

Ligne 16 La fin d'un programme en pascal est identifiée par un point après le mot clé `end`.