Département STPI Jeudi 12 avril 2012

I3 - Algorithmique

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

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 Question de cours (3 points)

1. Quelle est la différence entre itération déterministe et itération indéterministe ?
2. Qu'est ce que l'encapsulation ?
3. Quel est le rôle de la conception préliminaire ?

Solution proposée :
Voir le cours

2 Le jeu Sogo (8 points)

≪ Le Sogo est le nom donné en 1978 à la première édition française par la société Ravensburger d'un jeu qui existait depuis 1968 aux États-Unis d'Amérique sous le nom de Score four. Les premières versions du jeu datent de 1968 alors que le jeu Puissance 4 n'est apparu qu'en 1974.

Le Sogo est un jeu d'alignement en trois dimensions, avec un effet de gravité : une pièce ne peut être jouée à un niveau supérieur que si des pièces ont déjà été jouées à tous les niveaux inférieur au même endroit.

Le but du jeu est d'aligner 4 pièces de sa couleur le premier. L'alignement peut être vertical, horizontal ou suivre n'importe quelle diagonale. ≫ (Wikipédia).

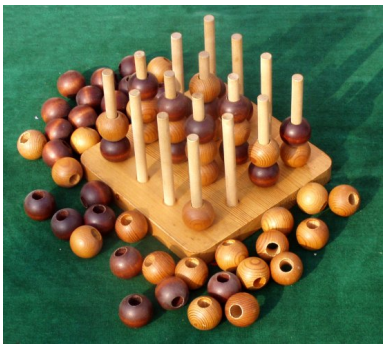


FIGURE 1 – Le jeu Sogo (source Wikipédia)

Comme nous l'avons vu en cours, on peut modéliser ce jeu de la façon suivante :
Constante LARGEUR = 4
Constante PROFONDEUR = 4
Constante HAUTEUR = 4
Type Contenu = {vide,pionNoir,pionBlanc}

1. Sauf les dictionnaires pour les étudiants non francophones

Type PlateauSogo **= tableau[***1..LARGEUR* **][** *1..PROFONDEUR*]**[***1..HAUTEUR*] **de** Contenu

1. Proposez la procédure `vider`, qui permet de vider entièrement un plateau du jeu Sogo.
2. Proposez la fonction `hauteur`, qui permet de calculer le nombre de pions verticaux présents sur une colonne (identifiée par son abscisse et son ordonnée).
3. Proposez la procédure `jouerUnPion`, qui permet de jouer un pion sur une colonne (identifiée par son abscisse et son ordonnée). Cette colonne ne devant pas être totalement remplie.

Solution proposée :

procédure vider **(E/S** p : Plateau**)**

  Déclaration i,j,k : **Naturel**

debut

  pour i ←1 **à** LARGEUR **faire**
    pour j ←1 **à** PROFONDEUR **faire**
      pour k ←1 **à** HAUTEUR **faire**
        p[i][j][k] ← vide
      finpour
    finpour
  finpour

fin

fonction hauteur (p : Plateau, largeur : 1..LARGEUR,profondeur : 1..PROFONDEUR**) :** 0..HAUTEUR

  Déclaration i : 0..HAUTEUR

debut

  i ← 1
  tant que p[largeur,profondeur,i]≠vide et i<=HAUTEUR **faire**
    i ← i+1
  fintantque
  retourner i-1

fin

procédure jouerUnPion **(E/S** p : Plateau,**E** largeur : 1..LARGEUR, profondeur : 1..PROFONDEUR, pion : pionNoir..pionBlanc**)**

  ⌊**précondition(s)** hauteur(p,largeur,profondeur)<HAUTEUR

debut

  p[largeur,profondeur,hauteur(p,largeur,profondeur)+1] ← pion

fin

3 Une analyse descendante (9 points)

Soit l'analyse descendante présentée par la figure 2 qui permet de rechercher la position d'une chaîne de caractères dans une autre chaîne indépendemment de la casse (d'où le suffixe IC à l'opération `positionSousChaineIC`), c'est-à-dire que l'on ne fait pas de distinction entre majuscule et minuscule.

Pour résoudre ce problème il faut pouvoir :

- obtenir la longueur d'une chaîne de caractères ;
- obtenir la sous-chaîne d'une chaîne en précisant l'indice de départ de cette sous-chaîne et sa longueur (le premier caractère d'une sous-chaîne à l'indice 1) ;
- savoir si deux chaînes de caractères sont égales indépendemment de la casse.

L'opération `positionSousChaineIC` retournera la première position de la chaîne recherchée dans la chaîne si cette première est présente, 0 sinon.

Par exemple :

- `positionSousChaine("AbCdEfGh","cDE")` retournera la valeur 3 ;

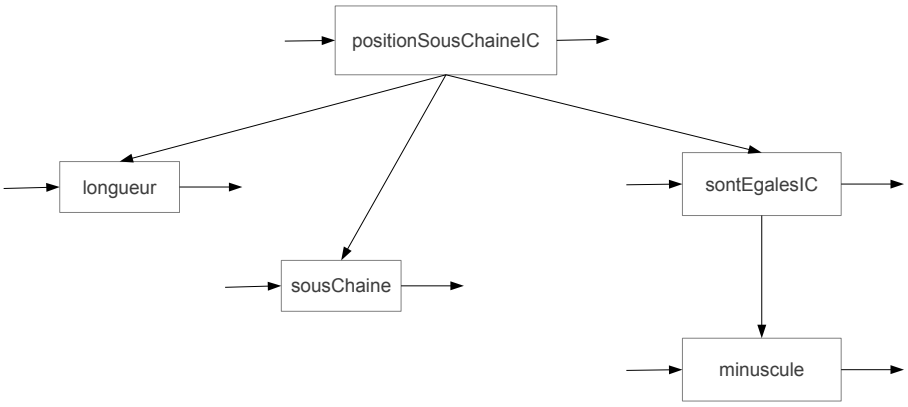
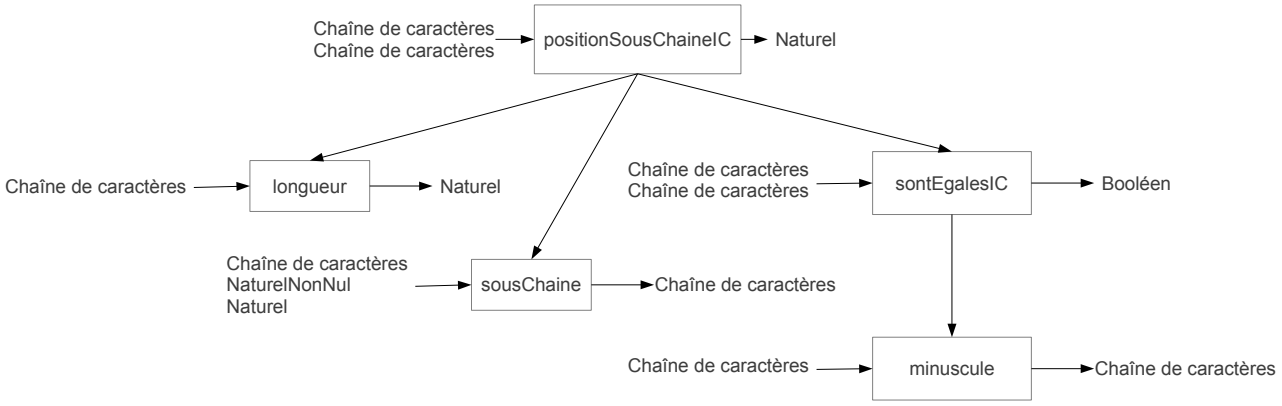


FIGURE 2 – Une analyse descendante

- `positionSousChaine("AbCdEfGh","abc")` retournera la valeur 1 ;
- `positionSousChaine("AbCdEfGh","xyz")` retournera la valeur 0.

1. Complétez l'analyse descendante en précisant les types de données en entrée et en sortie.
2. Donnez les signatures complètes (avec préconditions si nécessaire) des sous-programmes (fonctions ou procédures) correspondant aux opérations de l'analyse descendante.
3. Donnez l'algorithme du sous-programme correspondant à l'opération `positionSousChaineIC` (vous ne donnerez pas les algorithmes des autres opérations).
4. Donnez le code Pascal de ce sous programme en ayant au préalable déclarer les autres sous-programmes.

Solution proposée :



fonction positionIC (chaine, chaineARechercher : **Chaine de caracteres) : Naturel**

⌊**précondition(s)** longueur(chaineARechercher)≤longueur(chaine))

fonction longueur (chaine : **Chaine de caracteres) : Naturel**
fonction sousChaine (chaine : **Chaine de caracteres**, position : **NaturelNonNul**, long : **Naturel) : Chaine de caracteres**

⌊**précondition(s)** position+long≤longueur(chaine)

fonction sontEgalesIC (chaine1, chaine2 : **Chaine de caracteres) : Booleen**
fonction minuscule (chaine : **Chaine de caracteres) : Chaine de caracteres**

fonction positionIC (chaine, chaineARechercher : **Chaine de caracteres) : Naturel**

⌊**précondition(s)** longueur(chaineARechercher)≤longueur(chaine))

Déclaration i : **Naturel**

debut
i ← 1
tant que i+longueur(chaineARechercher)≤longueur(chaine) et non sontEgalesIC(sousChaine(chaine,i, longueur(chaineARechercher)),chaineARechercher) **faire**
i ← i+1
fintantque
si i+longueur(chaineARechercher)>longueur(chaine) **alors**
i ← 0
finsi
retourner i
fin

```
function longueur(s : String) : Word; forward;
function sousChaine(s : String; deb, long : Word) : String; forward;
function sontEgalesIC(s1,s2 : String) : Boolean; forward;

function positionSousChaineIC(s1,s2 : String) : Word;
begin
   positionSousChaineIC := 1;
   while (positionSousChaineIC+longueur(s2)<=longueur(s1))
         and not sontEgalesIC(sousChaine(s1,positionSousChaineIC,longueur(s2)),s2)
      positionSousChaineIC := positionSousChaineIC+1;
   if positionSousChaineIC+longueur(s2)>longueur(s1) then
      positionSousChaineIC:=0
end;
```