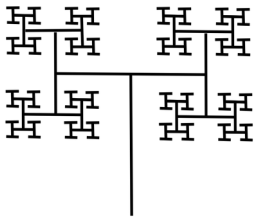
DS - Algorithmes et Structures de Données

Vendredi 12 Janvier 2024

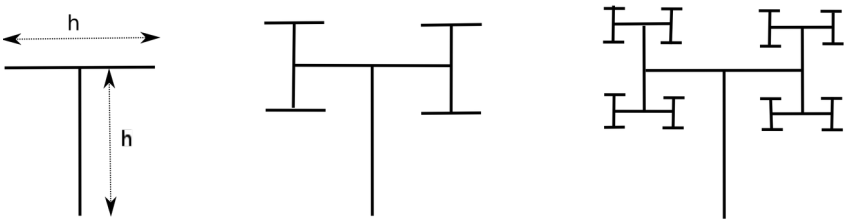
Durée 3H – Cours et TD NON autorisés

1. Fractale - 5 pts

On souhaite dessiner l'arbre suivant (de hauteur h=40 et de niveau n=4) :



Voici les différentes étapes pour n=1, n=2 et n=3, h est divisée par 2 à chaque étape :



On dispose d'une procédure `trace-T(E t,x,y,dir:entier)` qui trace un « T » de taille `t` (en pixels) dont la verticale est égale à l'horizontale, à partir du point de coordonnées `(x,y)` et dans la direction `dir` (= 1 si l'arbre est dirigé vers le haut et = -1 si vers le bas). On considère que l'origine des coordonnées est en haut à gauche de l'écran.

Ecrire en pseudo-langage la **procédure récursive** `dessine-arbre(E h,n,x,y,d:entier)` qui permet de dessiner un arbre de hauteur `h` et de niveau `n` à partir du point de coordonnées `(x,y)` et dans la direction `d`. Pour notre exemple, l'appel se fera avec `dessine-arbre(40,4,500,500,1)`.

```
Procédure dessine-arbre(E h,n,x,y,d : entier)
Début
Si n≥1

   Alors trace-T(h,x,y,d)
         h ← h div 2
          dessine-arbre(h,n-1,x-h,y-2*h*d,-1)
         dessine-arbre(h,n-1,x-h,y-2*h*d,1)
         dessine-arbre(h,n-1,x+h,y-2*h*d,-1)
         dessine-arbre(h,n-1,x+h,y-2*h*d,1)
FinSi
Fin
```

2. Tri par sélection d'une liste chaînée - 5 pts

On souhaite réaliser un tri par sélection sur une liste `l` de type `liste` : on prend le plus grand élément de `l` qu'on insère au début de la liste en construction. On veut garder la cellule « supprimée » de `l` (sans utiliser `insérer` ni `supprimer` du TAD du cours et en procédant par modification des chaînages des éléments de la liste sans utiliser `alouer`, ni `récupérer`).

```
Type   liste : ^cellule
       cellule : Enregistrement
                    val : entier
                   suiv : liste
                 FinEnregistrement
```

2.1. Ecrire une procédure `retourner-max(E/S l : liste, S pmax : ^cellule)` qui retourne en sortie un pointeur `pmax` sur le plus grand élément de la liste `l` et effectue la modification des chainages dans `l` pour « supprimer » cet élément (sans appeler `récupérer`).

On suppose l≠nil

```
Procédure retourner-max(E/S l : liste, S pmax : ^cellule)
Var max: entier
    p,q : liste
Début
max←l^.val
pmax←l
p←l
TantQue (p^.suiv≠nil) Faire
       Si max<p^.suiv^.val
              alors max←p^.suiv^.val
                    pmax←p^.suiv
                    q←p
       Finsi
       p←p^.suiv
FinTanQue
Si pmax = l
       alors l←l^.suiv
       Sinon q^.suiv←pmax^.suiv
FinSi
Fin
```

2.2. Ecrire une procédure `trier-selection(E/S l : liste)` qui permet de trier la liste `l` (elle appelle la procédure `retourner-max`).

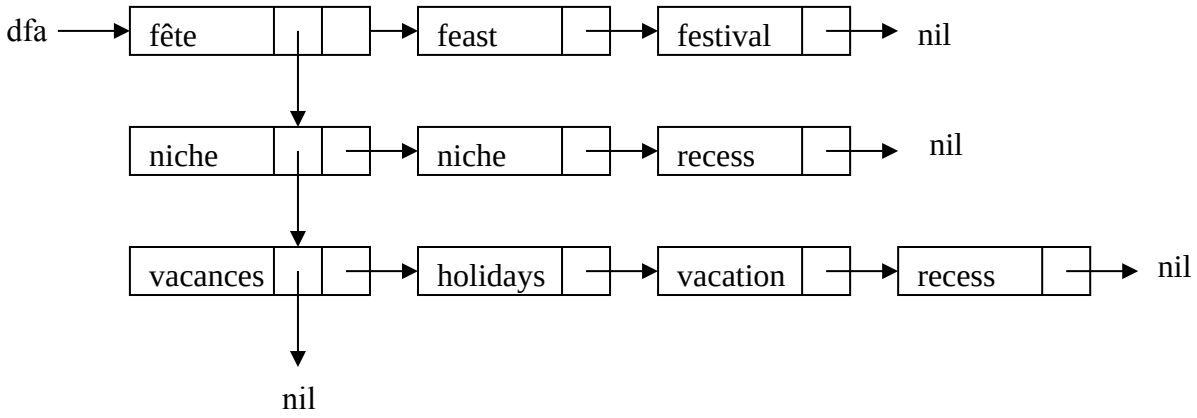
```
Procédure trier-selection(E/S l : liste)
Var p : liste
Début
p←nil
TantQue l≠nil Faire
       retourner-max(l,pmax)
       {insertion de la cellule pointée par pmax en tête de p}
       pmax^.suiv←p
       p←pmax
FinTantQue
l←p
Fin
```

3. Dictionnaire français-anglais - 10 pts

On souhaite manipuler des dictionnaires de traduction d'une langue en une autre. Les mots d'entrée sont triés dans l'ordre alphabétique et peuvent avoir plusieurs traductions (dans un ordre quelconque). Un tel dictionnaire peut être représenté par la structure de données suivante :

```
Type   traduc: Enregistrement
                      valtraduc : chaîne
                      traducsuiv : ^traduc
              FinEnregistrement
       mot : Enregistrement
                      valmot : chaîne
                       motsuiv : ^mot
                      listetraduc : ^traduc
              FinEnregistrement
       dico : ^mot
```

Exemple pour un dictionnaire français/anglais (`dfa` de type `dico` est trié sur `valmot`) :



3.1. Ecrire une procédure `donner-traduc` qui affiche à l'écran les toutes les traductions d'un mot `m` donné à partir du dictionnaire `d`.

```
Procédure donner-traduc (E d : dico, m : chaine)
Var p,q : ^traduc
Début
p←d
TantQue p≠nil et p^.valmot<m faire
    p←p^.motsuiv
FinTantQue
Si p≠nil et p^.valmot=m
       Alors q←p^.listetraduc
              TantQue q≠nil Faire
                     écrire(q^.valtraduc,' ')
                     q←q^.traducsuiv
              FinTantQue
       Sinon écrire('le mot n'existe pas')
Finsi
Fin
```

3.2. Ecrire une procédure `insérer-mot` qui insère un mot `m` (sans traduction) à la bonne place dans le dictionnaire `d`, et renvoie un pointeur `pm` sur la cellule du mot. Si le mot existe déjà, l'insertion n'est pas faite et le pointeur `pm` est retourné. Attention : `d` peut être vide.

```
Procédure insérer-mot (E m : chaine, E/S d : dico, S pm :^mot)
Var p : dico
Début
Si d=nil ou m<p^.valmot {le dictionnaire est vide ou insertion en tête}
    Alors  pm←allouer(mot)
           pm^.valmot←m
           pm^.motsuiv←d
           pm^.listetraduc←nil
           d←pm
    Sinon  p←d
           Si m=p^.valmot  {le mot existe déjà en tête}
               Alors pm←p
               Sinon   {On cherche la place où insérer}
                     TantQue p^.suiv≠nil  et p^.suiv^.valmot<m faire
                           p←p^.motsuiv
                     FinTantQue
                     Si p^.suiv^.valmot=m      {le mot existe déjà}
                           Alors pm←p^.suiv
                           Sinon  pm←allouer(mot)      {on insère après la cellule
                                  pm^.valmot←m           pointée par p}
                                  pm^.listetraduc←nil
                                  pm^.motsuiv←p^.motsuiv
                                  p^.motsuiv←pm
                     FinSi
            FinSi
FinSi
Fin
```

3.3. Ecrire une procédure `insérer-traduc` qui insère une traduction `t` pour un mot `m` (présent ou non dans `d`), sans utiliser la procédure `insérer` du TAD du cours. Elle utilise `insérer-mot`.

```
Procédure insérer-traduc (E m, t : chaine, E/S d : dico)
Var pm : dico
    r : ^traduc
Début
insère-mot(m,d,pm)
r←allouer(traduc)   {On insère la traduc en tête}
r^.valtraduc←t
r^.traducsuiv←pm^.listetraduc
pm^.listetraduc←r
Fin
```

3.5. Ecrire une procédure `inverser` qui construit le dictionnaire anglais/français `daf` à partir de celui français/anglais `dfa` (utiliser les procédures précédentes si besoin).

```
Procédure inverser(E dfa : dico, S daf : dico)
Var p : ^traduc
Début
daf←nil
Tantque dfa≠nil Faire
    p←dfa^.listetraduc
    TantQue p≠nil Faire
       insérer-traduc(dfa^.valmot,p^.valtraduc,daf)
        p←p^.traducsuiv
    FinTantQue
    dfa←dfa^.motsuiv
FinTantQue
Fin
```