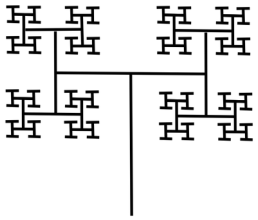
DS - Algorithmes et Structures de Données

Vendredi 12 Janvier 2024

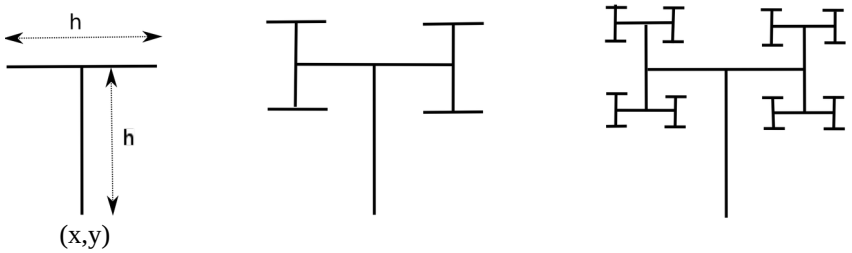
Durée 3H – Cours et TD NON autorisés

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)` (à la base du T) 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)`.

2. Tri d'une liste chaînée - 5 pts

On souhaite réaliser un tri par sélection sur une liste `l` de type `liste` : à chaque itération, on prend le plus grand élément de `l` qu'on insère au début de la liste en construction `templ`. On veut réutiliser la cellule retirée de `l` (sans utiliser `insérer` ni `supprimer` du TAD du cours) en procédant par modification des chaînages des éléments de la liste (sans utiliser `allouer`, 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 un pointeur `pmax` sur la cellule contenant le plus grand élément de la liste `l` et effectue la modification des chainages dans `l` pour retirer cet élément (sans appeler `récupérer`).

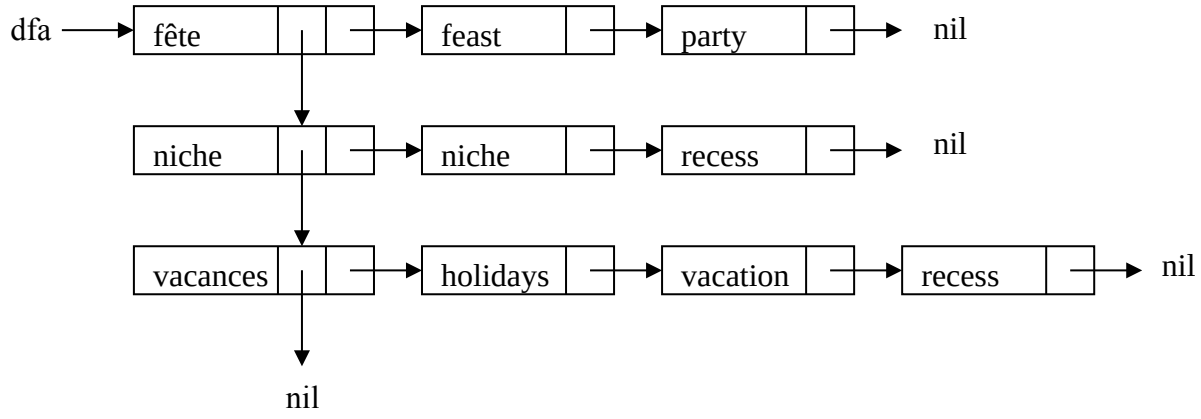
2.2. Ecrire une procédure `trier-selection (E/S l : liste)` qui permet de trier la liste `l` (elle appelle la procédure `retourner-max`). On pourra utiliser une variable temporaire locale `templ` pour construire la liste triée et à la fin de la procédure `l` recevra `templ`.

3. Dictionnaire - 10 pts

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

```
Type   celtraduc: Enregistrement
                    traduc : chaîne
                    suiv : ^celtraduc
              FinEnregistrement
       celmot : Enregistrement
                    mot : chaîne
                    suiv : ^celmot
                    ltraduc : ^celtraduc
              FinEnregistrement
       dico : ^celmot
```

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



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

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.

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`.

3.4. Dessiner le dictionnaire anglais/français correspondant à l'exemple donné.

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).