Algorithmes et Structures de Données

Mardi 8 Novembre 2022

Durée 1H30 – Cours et TD NON autorisés

1. Motus - 10 pts

On souhaite écrire un programme qui joue au jeu inspiré de MOTUS : retrouver un mot de 4 à 10 lettres en un minimum de coups !

Pour cela, on dispose d'un fichier texte comportant 100 mots permis, triés dans l'ordre alphabétique (un mot par ligne). L'utilisateur choisit un mot permis et le programme doit le deviner : la longueur du mot est donnée ainsi que la première lettre. A chaque étape, le programme propose un mot et l'utilisateur répond par un tableau de valeurs de la longueur du mot :

- 2 : la lettre correspondante est bien placée
- 1 : la lettre correspondante est présente mais mal placée
- 0 : la lettre correspondante n'est pas présente dans le mot

Exemple : mot à trouver = 'bonnet', mot proposé = 'bleues', réponse = 2 0 1 0 2 0

Le programme tient compte des réponses de l'utilisateur pour proposer de nouveaux mots candidats jusqu'à ce qu'il trouve le mot en question (il a gagné) ou qu'il atteigne le nombre maximal (par ex. 5) de propositions (il a perdu).

1.1. Ecrire en pseudo-langage une procédure `charger_tab` qui charge le fichier de mots (dont le nom est dans `nomf`) dans un tableau de mots `t` de taille effective `nbmots`.

```
Type tab_mots : tableau[1..100] de chaine

Procédure charger_tab (E nomf : chaine  S t : tab_mots, nbmots : entier)
```

1.2. Ecrire en pseudo-langage une procédure `créer_mots_candidats` qui, à partir de `t`, renvoie un tableau de mots candidats `t_cand` (mots de même longueur et commençant par la même lettre que le mot cherché `mot`) de taille effective `nb`. L'algorithme doit être optimal.

```
Procédure créer_mots_candidats (E t : tab_mots, n : entier, mot : chaine,
                                S t_cand : tab_mots, nb : entier)
```

1.3. Ecrire en pseudo-langage une fonction `correspondre` qui renvoie `vrai` ou `faux` selon si le mot `m1` correspond au mot `m2` (formé de lettres et de `'?'`), contient les lettres de `oui` (lettres présentes mais mal placées) et ne contient pas les lettres de `non` (lettres non présentes).

```
m1='bleues' ; m2='b???es' ; oui='l' ; non='rgt' ; m1 et m2 correspondent
m1='bleues' ; m2='br??es' ; oui='l' ; non='rgt' ; m1 et m2 ne correspondent pas
m1='brumes' ; m2='b???es' ; oui='u' ; non='m' ; m1 et m2 ne correspondent pas

Fonction correspondre (m1,m2,oui,non : chaine) : booléen
```

1.4. Ecrire en pseudo-langage une procédure `cherche_cand` qui cherche un mot `s` qui corresponde à `m`, `oui` et `non`, à partir du tableau de mots candidats `t_cand`.

```
Procédure chercher_cand (E t_cand : tab_mots, nb : entier, oui, non, m : chaine,
                         S s : chaine)
```

1.5. Ecrire en pseudo-langage le programme complet qui implémente l'interaction entre l'ordinateur et l'utilisateur jusqu'au succès ou l'échec du jeu.

2. Pile - 5 pts

```
Fonction Cantor(x,y : entier): entier
Si x=0 et y=0
      Alors res ← 0
      Sinon Si y=0
                  Alors res ← Cantor(0,x-1){@1} + 1
                  Sinon res ← Cantor(x+1,y-1){@2} + 1
            FinSi
FinSi
Retourner(res)
Fin
```

Simuler la pile pour l'appel `écrire(Cantor(3,2)){@0}`

3. Récursivité – 5 pts

On souhaite écrire à l'écran les chaines de caractères suivantes selon n :

Pour n=0 : `'0=0'`
n=1 : `'0=(0+0)'`
n=2 : `'0=((0+0)+(0+0))'`
n=3 : `'0=(((0+0)+(0+0))+((0+0)+(0+0)))'`
...

3.1. Expliquer le principe récursif.

3.2. Ecrire en pseudo-langage une procédure **récursive** qui effectue cet affichage.