

| | | |
|---|---|--------------------------|
| REPUBLIQUE TUNISIENNE MINISTRE DE L'EDUCATION ET DE LA FORMATION *** EXAMEN DU BACCALAUREAT SESSION 2008 (SP) | Sections : Math. + Tech. + Sc.Exp. | |
| | EPREUVE THEORIQUE D'INFORMATIQUE | |
| | SOLUTION | |
| | DUREE : 1 h | COEFFICIENT : 0.5 |

PARTIE I (8 points)**Exercice 1 : (3 points = 4 x 0,75)**

Compléter le tableau ci-dessous, par les déclarations Pascal adéquates :

- 0.25 / mot clé + 0.5 pour la suite de la déclaration

| Description | Déclaration en Pascal (Préciser le mot clé adéquat : CONST, TYPE, VAR, etc.) | Remarques |
|--|---|---|
| Une chaîne ch de 20 caractères au maximum. | Var ch : string[20] ; | |
| Un type saison contenant les identificateurs suivants : Automne, Hiver, Printemps, etc. | Type saison = (Automne, Hiver, Printemps, etc) ; | L'ordre est important |
| Un tableau V dont les indices sont de type caractère pouvant contenir 20 chaînes. | Var V : array['A'.. 'T'] of string ; | On acceptera toute combinaison équivalente de 20 éléments |
| Une constante message de valeur « Bonne chance ». | Const Message='Bonne chance' ; | |

Exercice 2 : (2 points = 8 x 0,25)

Soit la fonction Existe dont l'algorithme est donné ci-dessous :

0) DEF FN Existe (n: entier ; T : **tab** ; x : **réel**) : **booléen**

1) Trouve ← **faux**

2) i ← 0

3) Répéter

i ← **i + 1**

Si (T[i] = x) alors

Trouve ← vrai

FinSi

4) Jusqu'à (**Trouve**) ou (**i = n**)

5) **Existe** ← **Trouve**

6) Fin Existe



- On acceptera que le type de la fonction soit **chaîne** avec la séquence **5)** une structure conditionnelle cohérente.

Compléter l'algorithme de la fonction **Existe**, dont les paramètres sont x , n et T , qui permet de vérifier l'existence d'un élément x dans un tableau T de n réels.

Exercice 3 : (3 points)

Compléter les affectations suivantes par une valeur d'opérande ou d'opérateur permettant d'obtenir dans chacun des cas, la valeur de Y voulue :

| Affectations | Valeur de Y | |
|---|-------------|-------------------------------------|
| $Y := \text{round}(99,51) = \mathbf{100}$; | True | |
| $Y := (\text{upcase}('a') \text{ in } ['A' .. 'Z']) \text{ and } (1 \text{ in } [1..10])$; | True | Toute valeur comprise entre 1 et 10 |
| $Y := \text{length}('PASCAL') \bmod 4 = 2$; | True | |
| $Y := \text{random}(4) < 4$; | True | $Y := \text{random}(4) < 4$; |
| $Y := \text{pred}('D') = \text{chr}(\text{ord}('B') + 1)$; | True | |
| $Y := \text{copy}('informatique', 1, 4) < 'info'$; | False | |

NB. : Certaines affectations ont plusieurs solutions correctes.



Solution de la partie 2 : (12 points)

1) Structures données

(1.5 pt)

On évaluera les structures de données de base et non les compteurs (la chaîne à traiter, le conteneur des fréquences des lettres, la variable pour les caractères non alphabétiques)

La chaîne à traiter sera nommée ch.

Les résultats du problème peuvent être récupérées dans un tableau LET pour les lettres et dans une variable NLET pour les autres.

Le tableau LET est de type TLET qui est un tableau de 26 entiers et qui aura comme indices la plage des lettres alphabétiques majuscules "A" à "Z".

2) Analyse du programme principal et des modules

| Points clés de l'analyse | Barème |
|--|--------------------------|
| Programme principal (modularité + cohérence) | 1.5 |
| Saisie contrôlée de ch | 1 |
| Le comptage | 2 + 0.5 (initialisation) |
| Premier affichage | 0.25 |
| Deuxième affichage | 1 |
| Troisième affichage | 0.25 |
| TDO et TNT | 0.5 |

| | NOM = nbre_lettres | |
|---|--|----------|
| S | L.D.E. | O.U. |
| 3 | Résultat = (Ecrire(" Votre texte comporte ", longueur(ch), " caractères dont : "), | afficher |
| 2 | PROC afficher(LET), écrire(" et " , NLET, " caractères non alphabétiques.") | decompte |
| 1 | (LET, NLET) = PROC decompte(ch, LET, NLET) | LET |
| | ch = REPETER | NLE |
| | DONNEE ("Chaîne à traiter :") | Ch |
| | JUSQU'A (longueur (ch) dans [1..100]) | |
| 4 | Fin nbre_lettres | |

Tableau de déclaration des nouveaux types

| TYPE |
|--|
| TLET tableau de 26 entiers et dont les indices sont "A", "B", ... , "Z" |

Tableau de déclaration des objets

| NOM | TYPE | ROLE |
|----------|-------------|--|
| afficher | procédure | - permet d'afficher les éléments du tableau avec les commentaires relatifs |
| decompte | procédure | - permet de déterminer la décomposition de la chaîne |
| LET | TLET | - ses éléments comportent respectivement le nombre d'une lettre de la chaîne ch. |
| NLET | Entier | - le nombre de caractères non alphabétiques |
| ch | Chaine[100] | - chaîne à traiter |

3) Analyse des modules

analyse du module **afficher**



| | DEFPROC afficher(TL : TLET) | |
|---|--|------|
| S | L.D.E. | O.U. |
| 1 | Résultat = affiche_lettres affiche_lettres = [] Pour c de "A" à "Z" faire [] Si (TL[c] ≠ 0) alors Ecrire(TL[c], " fois la lettre ", c) FinSi | c |
| 2 | FinPour Fin afficher | |

Tableau de déclaration des objets locaux

| NOM | TYPE | ROLE |
|-----|-----------|----------------------------------|
| c | caractère | compteur et en même temps indice |

Analyse du module **decompte**

| | DEFPROC decompte(cht : chaîne[100] ; VAR TL : TLET ; VAR NLT : entier) | |
|---|---|---------------------|
| S | L.D.E. | O.U. |
| 1 | Résultat = (TL, NLT) (TL, NLT) = [l ← Long(cht), NLT ← 0, PROC init(TL),] Pour i de 1 à l Faire [c ← Majus(cht[i])] Si (c dans ["A".."Z"]) alors TL[c] ← TL[c] + 1 Sinon NLT ← NLT + 1 FinSi | l init i c |
| 2 | FinPour Fin decompte | |

Tableau de déclaration des objets locaux

| NOM | TYPE | ROLE |
|------|------------------------------------|---|
| l | entier | longueur de la chaîne |
| init | procédure locale à decompte | sert à initialiser à 0 les éléments du tableau TL |
| i | entier | compteur |
| c | caractère | compteur |

Analyse du module **init**

| | DEFPROC init(VAR TL : TLET) | |
|---|--|------|
| S | L.D.E. | O.U. |
| 1 | Résultat = TL TL = [] Pour c de "A" à "Z" Faire TL[c] ← 0 FinPour | c |
| 2 | Fin init | |



Tableau de déclaration des objets locaux

| NOM | TYPE | ROLE |
|-----|-----------|----------------------------------|
| c | caractère | compteur et en même temps indice |

4) Les algorithmes**Algorithme du programme principal (1pt)**

- 0) Debut nbre_lettres
 1) REPETER (Saisie contrôlée 0.5 pt)
 DONNEE ("Chaîne à traiter :")
 JUSQU'A (longueur (ch) dans [1..100])
 2) PROCdécompte(ch, LET, NLET)
 3) Ecrire(" Votre texte comporte ", longueur(ch), " caractères dont : "), PROC
 afficher(LET), écrire(" et ", NLET, " caractères non alphabétiques.")
 4) Fin nbre_lettres

Algorithmes des modules (2,5pt -0.25 / faute)**Algorithme du module afficher (1pt)**

- 0) DEFPROCafficher(TL : TLET)
 1) Pour c de "A" à "Z" faire
 Si (TL[c] ≠ 0) alors
 Ecrire(TL[c], " fois la lettre ", c)
 FinSi
 FinPour
 2) Fin afficher

Algorithme du module décompte et init (1pt)

- 0) DEFPROCdecompte(cht : chaîne[100] ; **VAR** TL : TLET ; **VAR** NLT : entier)
 1) [l ← Long(cht), PROC init(LT)] Pour i de 1 à l Faire
 [c ← Majus(cht[i])] Si (c dans ["A".."Z"]) alors
 TL[c] ← TL[c] + 1
 Sinon
 NLT ← NLT + 1
 FinSi
 FinPour
 2) Fin decompte

Algorithme du module init

- 0) DEFPROC init(**VAR** TL : TLET)
 1) Pour c de "A" à "Z" Faire



$TL[c] \leftarrow 0$

FinPour

2) Fin init

