


ALGORITHME

Exercices et Proposition de Corrigé

	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 1
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

1. EXERCICE : CALCULER LE NOMBRE DE JEUNES.	3
2. EXERCICE : CALCULER LE NOMBRE DE JEUNES, DE MOYENS ET DE VIEUX.	4
3. EXERCICE : COMPTER UNE LETTRE DANS UNE PHRASE.....	6
4. EXERCICE : COMPTER LE NOMBRE DE CARACTERES DANS UNE PHRASE.....	8
5. EXERCICE : COMPTER LES OCCURRENCES DE DEUX LETTRES SUCCESSIVES DANS UNE PHRASE.	9
6. EXERCICE : RECHERCHER UNE SYMETRIQUE DANS UNE CHAINE DE CARACTERES.....	11
7. EXERCICE : TRIER UN TABLEAU D'ENTIERES PAR LA METHODE DE TRI PAR REMONTEE DES BULLES.....	13
8. EXERCICE : RECHERCHER PAR DICHOTOMIE UN ELEMENT D'UNE TABLE CLASSEE.....	21
9. EXERCICE : DONNER LA VALEUR DU X-IEME BIT D'UN ENTIER.....	25
10. EXERCICE : FAIRE LE ET LOGIQUE DE DEUX ENTIERES.....	28
11. EXERCICE : CODER UNE PHRASE EN UNE AUTRE EN CODE ASCII.	30
12. EXERCICE : CHERCHER UN MOT DANS UNE PHRASE.....	35
13. EXERCICE : RECOPIER UNE PHRASE EN INVERSANT CHAQUE MOT.	44
14. EXERCICE : JUSTIFIER UNE PHRASE.	53
15. EXERCICE : CALCULER LA SOMME DE NOMBRES EN BASE QUELCONQUE.	63
16. EXERCICE : VERIFIER LA SYNTAXE D'UNE COMMANDE.	72
17. EXERCICE : STRUCTURER DES DONNEES ASSOCIEES A DES PIECES.	86
18. EXERCICE : MANIPULER UNE PILE D'ENTIERES GEREES AVEC UN TABLEAU ET UN INDICE.....	87
19. EXERCICE : MANIPULER UNE TABLE GEREES PAR HASH-CODE	91
20. EXERCICE : GERER UNE LISTE DE NOMS CLASSES ALPHABETIQUEMENT.....	94
21. EXERCICE : CREER UNE LISTE DYNAMIQUE GEREES EN FIFO.....	102
22. EXERCICE : GERER UNE LISTE DYNAMIQUEMENT.....	107
23. EXERCICE : FICHIER SEQUENTIEL DE NOMBRES CONSERVES PAR ORDRE CROISSANT.....	113
24. EXERCICE : GESTION D'UN FICHIER DE SALAIRES.....	117

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 2
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

1. EXERCICE : calculer le nombre de jeunes.

Il s'agit de dénombrer toutes les personnes d'âge inférieur strictement à vingt ans parmi un échantillon donné de vingt personnes. Les personnes saisissent leur âge sur le clavier.

Donnez l'algorithme correspondant.

1.1 JEUX D'ESSAI : calculer le nombre de jeunes.

1) pas de jeunes (>20)

45 35 65 76 34 32 31 46 57 68 75 43 53 36 31 46 68 59 30 43

2) pas de non_jeunes (<20)

15 5 5 6 4 2 11 16 7 8 7 3 13 16 11 18 8 9 19 3

3) des jeunes et des non-jeunes (20 nombres)

45 35 65 76 34 20 20 30 30 30 20 20 30 20 30 20 20 8 15 23

1.2 SOLUTION : calculer le nombre de jeunes.

Programme nombre_de_jeunes

// Ce programme compte le nombre de personnes d'un âge inférieur à une limite
// dans un échantillon de personnes de nombre donné

Constantes Nbgens = 20 // nombre de personnes à interroger
Limite = 20 // âge à partir duquel les personnes ne sont plus jeunes

Variables Nbgensint : entier // nombre de personnes déjà interrogées
Nbjeunes : entier // nombre de personnes dont l'âge est
// inférieur à la limite
Age : entier // âge de la personne interrogée

Début // initialisation

Nbgensint := 0
Nbjeunes := 0

// interrogation des personnes une à une

Tantque Nbgensint < Nbgens **Faire** // arrêt quand toutes les personnes ont été interrogées
Ecrire ('Entrez votre âge, s.v.p.: ')
Lire (Age) // Interrogation d'une personne
Nbgensint := Nbgensint + 1

Si Age < Limite **Alors** // c'est un jeune
Nbjeunes := Nbjeunes + 1

Finsi

Fintantque

// Résultat du sondage

Ecrire ('le nombre de jeunes sur cet échantillon de 'Nbgens,' personnes est de : 'Nbjeunes')

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 3
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

2. EXERCICE : calculer le nombre de jeunes, de moyens et de vieux.

Il s'agit de dénombrer les personnes d'âge inférieur strictement à 20 ans, les personnes d'âge supérieur strictement à 40 ans et celles dont l'âge est compris entre 20 ans et 40 ans (20 ans et 40 ans y compris).
Le comptage est arrêté dès la saisie d'un centenaire.

Donnez l'algorithme correspondant qui affiche les résultats.

2.1 JEUX D'ESSAI : *calculer le nombre de jeunes, de moyens et de vieux.*

1) *juste un centenaire*

105

2) *juste un jeune centenaire*

100

3) *tests des limites*

20 20 20 40 40 40 100

4) *test global*

17 18 19 20 21 22 37 38 39 40 41 42 43 44 102

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 4
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

2.2 SOLUTION : calculer le nombre de jeunes, de moyens et de vieux.

Programme nombre_jeunes_moyens_vieux

// ce programme compte le nombre de personnes d'un âge inférieur à une limite basse, le nombre de
 // personnes d'un âge supérieur à une limite haute, le nombre de personnes d'un âge compris entre
 // ces deux limites (limites comprises), jusqu'à rencontrer un centenaire, le centenaire sera compté.
 // dans les personnes à âge supérieur à la limite haute

Constantes Limbasse = 20 // limite basse pour compter les moyens
 Limhaute = 40 // limite haute pour compter les moyens
 Terminal = 100 // âge terminant le comptage

Variables Nbjeunes : entier // nombre de personnes jeunes comptées
 Nbmoysens : entier // nombre de personnes entre les deux âges
 Nbviex : entier // nombre de personnes "âgées"
 Age : entier // âge de la personne interrogée

Début // initialisation des compteurs

Nbjeunes := 0
 Nbmoysens := 0
 Nbviex := 0

// interrogation des personnes une à une

Répéter

Ecrire ('Entrez votre âge, s.v.p. :')

Lire (Age)

Si Age < Limbasse **Alors** // il s'agit d'un jeune
 Nbjeunes := Nbjeunes + 1

Sinon

Si Age > Limhaute **Alors** // il s'agit d'un vieux
 Nbviex := Nbviex + 1

Sinon // il s'agit d'un moyen
 Nbmoysens := Nbmoysens + 1

Finsi

Finsi

Jusqu'à Age >= Terminal // continuation jusqu'à la rencontre du centenaire
 // traitement d'au moins une personne : le centenaire.

// résultat du sondage

Ecrire ('Le nombre de jeunes est ',Nbjeunes)

Ecrire ('Le nombre de moyens est ',Nbmoysens)

Ecrire ('Le nombre de vieux est ',Nbviex)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 5
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

3. EXERCICE : compter une lettre dans une phrase.

Soit une chaîne de caractères terminée par le caractère '.'. Donnez l'algorithme d'un programme qui compte le nombre occurrences d'une lettre donnée ('a' par exemple) dans cette chaîne.

3.1 JEUX D'ESSAI : *compter une lettre dans une phrase.*

1) pas de lettre

lettre 'a' terminateur '.' phrase '.' résultat 0

2) pas de bonne lettre

lettre 'a' terminateur '.' phrase 'toto.' résultat 0

3) que des bonnes lettres

lettre 'a' terminateur '.' phrase 'aaaa.' résultat 4

4) mixtes

lettre 'a' terminateur '.' phrase 'zazie dans le métro.' résultat 2

5) la lettre cherchée est le terminateur

lettre '.' terminateur '.' phrase 'zazie dans le métro.' résultat 0

Remarque : le terminateur ne fait pas partie des caractères à traiter
le terminateur est présent dans la phrase, et nous nous préoccupons pas de le vérifier dans le cadre de ces exercices.

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 6
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

3.2 SOLUTION : compter une lettre dans une phrase.

Programme compter_une_lettre

// Ce programme compte les occurrences d'un caractère donné, ici 'a', dans une chaîne de caractères terminée par un caractère terminateur, ici le '.'

Constantes

carcible	= 'a'	// caractère dont les occurrences sont comptées
carterm	= '.'	// caractère indiquant la terminaison de la chaîne
taille	= 80	// nombre maximum de caractères dans la chaîne

Types

chaîne	= tableau [taille] de caractères	// type des chaînes de caractères traitées
--------	--	--

Variables

phrase	: chaîne	// phrase dans laquelle les carcibles sont comptés
i	: entier	// indice de parcours de la phrase
compteur	: entier	// compteur des carcibles dans la phrase

Début // initialisation des compteurs et lecture de la phrase

Ecrire ('donnez une phrase terminée par un caractère ',carterm)

Lire (phrase)

i := 1

compteur := 0

// parcours de la phrase avec recherche des caractères carcibles
// il y a 0 ou plusieurs caractères à parcourir

Tantque (phrase [i] <> carterm) **faire** // arrêt sur le caractère terminateur

Si phrase [i] = carcible **Alors**
compteur := compteur + 1 // un caractère cible trouvé

Finsi

i := i + 1 // passage au caractère suivant

Fintantque

// résultat du parcours

Ecrire ('le nombre de caractères ', carcible ', dans la phrase est : ',compteur)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 7
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

4. EXERCICE : compter le nombre de caractères dans une phrase.

Soit une chaîne de caractères terminée par le caractère '.'.

Donnez l'algorithme d'une procédure qui compte la longueur de cette chaîne ('.' non compris).

4.1 JEUX D'ESSAI : compter le nombre de caractères dans une phrase.

1) pas de lettre

terminateur '.' phrase '.'

résultat 0

2) des lettres avant la fin

terminateur '.' phrase 'toto.'

résultat 4

Remarque : le terminateur ne fait pas partie des caractères à traiter

4.2 SOLUTION : compter le nombre de caractères dans une phrase.

Procédure compter_caractère (Entrée phrase : chaîne , Entrée carfin : caractère ,
Sortie nbcar : entier)

// cette procédure compte le nombre de caractères dans une chaîne de caractères
// terminée par un caractère terminateur, ici le '.'

// phrase est la chaîne de caractères à traiter
// carfin est le caractère terminateur de la chaîne à traiter
// nbcar est le nombre de caractères trouvés

Variables i : entier // indice de parcours du texte

Début // initialisation de l'indice de parcours de la phrase

i := 1

// parcours de la phrase avec calcul du nombre de caractères
// il y a 0 ou plusieurs caractères à parcourir

Tantque (phrase [i] <> carterm) **faire** // arrêt sur le caractère terminateur

i := i + 1 // passage au caractère suivant

Fintantque
// résultat du parcours

nbcar := i - 1

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 8
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

5. EXERCICE : compter les occurrences de deux lettres successives dans une phrase.

Donnez l'algorithme d'une procédure qui, pour une chaîne donnée terminée par un caractère terminateur donné et pour deux lettres données ('l','e' par exemple), compte le nombre occurrences de ces deux lettres successives dans la chaîne.

Fournissez un programme de test de votre procédure.

5.1 JEUX D'ESSAI : compter les occurrences de deux lettres successives dans une phrase.

1) phrase vide					
couple 'ab'	final '.'	phrase	'.'	résultat	0
2) pas de couple					
couple 'ab'	final '.'	phrase	'coucou.'	résultat	0
3) la moitié du couple					
couple 'ab'	final '.'	phrase	'a.'	résultat	0
4) plusieurs couples					
couple 'ab'	final '.'	phrase	'ababab.'	résultat	3
5) plusieurs couples et un peu plus					
couple 'ab'	final '.'	phrase	'abababa.'	résultat	3
6) une phrase mixte					
couple 'ab'	final '.'	phrase	'la bête l'abbé et l'abreuvoir.'	résultat	2
7) couple de lettres identiques					
couple 'ee'	final '.'	phrase	'creee.'	résultat	1
8) couple avec un terminateur					
couple 'a.'	final '.'	phrase	'aa.'	résultat	0
8) couple de terminateurs					
couple '..'	final '.'	phrase	'.'	résultat	0

5.2 SOLUTION : compter les occurrences de deux lettres successives dans une phrase.

Procédure compter_occur_succ (**Entrée** texte : chaîne , **Entrée** carfin : caractère , **Entrée** lettre1 : caractère , **Entrée** lettre2 : caractère , **Sortie** nbcouple : entier)

// Cette procédure compte le nombre d'occurrences de deux lettres
// successives dans une chaîne de caractère. Si lettre1 = lettre2,
// 3 occurrences de lettre1 ne comptent que pour 1 occurrence du couple.
// 4 occurrences de lettre1 comptent pour 2 occurrences du couple.
// le point final ne fait pas partie de la chaîne pour les recherches

// texte est la chaîne de caractères à traiter
// carfin est le caractère terminateur de la chaîne à traiter
// lettre1 est le premier caractère du couple de lettres à chercher
// lettre2 est le second caractère du couple de lettres à chercher
// nbcouple est le nombre de couples de lettres successives trouvées

Variables i : entier // indice de parcours du texte

Début // initialisation du nombre de couples et de l'indice

nbcouple := 0
i := 1

// parcours de la chaîne à la recherche de couples

Tantque texte [i] <> carfin **Faire** // arrêt sur le caractère terminateur du texte

Si (texte [i] = lettre1) **Alors**

Si ((texte [i + 1] = lettre2) et (texte [i + 1] <> carfin))

nbcouple := nbcouple + 1
// il faut écarter le couple trouvé
i := i + 1

Finsi

Finsi

i := i + 1 // passage au caractère suivant

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 10
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

6. EXERCICE : rechercher une symétrie dans une chaîne de caractères.

Un palindrome est une chaîne de caractères que l'on peut lire identiquement de droite à gauche , et de gauche à droite.

Par exemple :

AA.
38783.
LAVAL.
LAVAL A ETE A LAVAL.

Soit une chaîne de caractères terminée par un point. Ecrivez l'algorithme d'un programme permettant d'affirmer si cette phrase est ou non un palindrome.

6.1 JEUX D'ESSAI : rechercher une symétrie dans une chaîne de caractères.

phrase

résultat

‘.’
‘a.’
‘aa.’
‘aba.’
‘acb.’
‘aacba.’
‘aacab.’

c'est un palindrome
c'est un palindrome
c'est un palindrome
c'est un palindrome
ce n'est pas un palindrome
ce n'est pas un palindrome
ce n'est pas un palindrome

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 11
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

6.2 SOLUTION : *rechercher une symétrie dans une chaîne de caractères.*

<u>Programme</u>	faux_palindrome
-------------------------	-----------------

```
// détermination si une chaîne de caractères, terminée par un point final,  
// présente un caractère de symétrie.
```

Constantes taille = 80 // nombre maximum de caractères dans la chaîne
 carterm = '.' // caractère indiquant la terminaison de la chaîne

Types chaîne = **tableau** [taille] **de caractères**
// type des chaînes de caractères traitées

<u>Variables</u>	phrase	:	chaîne	// phrase dont on va déterminer la symétrie
	i	:	<u>entier</u>	// indice de parcours de la phrase par le début
	j	:	<u>entier</u>	// indice de parcours de la phrase par la fin

Début // saisie de la phrase

Ecrire ('donnez une phrase terminée par un caractère ',carterm)

Lire (phrase) // calcul de la longueur de la phrase

$$j := 1$$

Tantque phrase [j] <> carterm
j := j + 1

Faire // arrêt sur le caractère terminateur

Fintantque

```
j := j - 1 // parcours de la phrase par les deux bouts pour ne pas compter le caractère terminateur
```

$$i := 1$$

Tantque (i < j) **et** (phrase [i] = phrase [j]) **Faire** // arrêt quand les indices se croisent ou
// quand il n'y a pas de symétrie

$$i := i + 1$$
$$j := j - 1$$

Fintantque

```
// affichage du résultat de la recherche
```

Si $i \geq j$ **Alors** // fin du parcours, il y a symétrie

Ecrire ('c'est un palindrome')

Sinon

Ecrire ('ce n'est pas un palindrome')

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 12
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

7. EXERCICE : trier un tableau d'entiers par la méthode de tri par remontée des bulles.

Principe : on parcourt le tableau de valeurs en comparant les éléments deux à deux. Si le plus grand est avant le suivant, on inverse les éléments dans le tableau.

On fait autant de parcours du tableau que nécessaire pour que le tableau soit trié (et qu'on s'en rende compte).

Exemple : Etat du tableau pendant le parcours :

premier parcours

9	1	1	1
1	9	4	4
4	4	9	2
2	2	2	9

en grisé éléments comparés deux à deux

il y a eu des inversions pendant le premier parcours

deuxième parcours

1	1	1	1
4	4	2	2
2	2	4	4
9	9	9	9

en grisé éléments comparés deux à deux

il y a eu des inversions pendant le deuxième parcours

troisième parcours

1	1	1	1
2	2	2	2
4	4	4	4
9	9	9	9

en grisé éléments comparés deux à deux

il n'y a pas eu d'inversions pendant le troisième parcours

Donnez l'algorithme de la procédure paramétrée correspondant à ce tri.

Faites votre algorithme le plus simple possible pour qu'il fasse le traitement tel qu'il est proposé.

Trouvez ensuite un algorithme qui respecte cette méthode de tri, mais qui l'optimise. Il ne va trier que ce qui est nécessaire, mais toujours selon la même méthode.

Optimisez progressivement, il y a 3 optimisations possibles.

7.1 SOLUTION : trier un tableau d'entiers par la méthode de tri par remontée des bulles.(programme)

Programme test_tri_bulles

// Ce programme teste le tri d'un tableau d'entiers par la méthode de remontée des bulles.

Constantes n = 80 // nombre maximum de nombres dans le tableau

Types table = tableau [n] de entier // type des tables d'entiers que l'on peut trier

Variables
 tabent : table // table d'entiers que l'on va trier
 taille : entier // longueur utile de la table d'entiers
 cpt : entier // compte le nombre de saisies d'entiers effectuées

Procédure tri_bulles (Entrée Sortie tab : table , Entrée lgutile : entier)

// Cette procédure trie le tableau d'entiers sur sa longueur utile, selon la méthode de remontée des bulles

// tab est : en entrée le tableau d'entiers à trier.
 // en sortie le tableau d'entiers triés.
 // lgutile est la longueur utile du tableau d'entiers.

Début // saisie de la table d'entiers à traiter et de sa longueur utile

Ecrire ('donnez le nombre d'entiers à entrer dans le tableau : ')

Lire (taille)

cpt := 0

Tantque cpt < taille **Faire**

cpt := cpt + 1 // saisie d'un nouveau nombre de la table

Ecrire ('donnez l'entier numéro ',cpt,' de la table')

Lire (tabent [cpt])

Fintantque

// appel de la procédure qui fait tout le travail

tri_bulles (tabent, taille)

// résultat du tri

Ecrire ('Voici le tableau trié: ')

cpt := 0

Tantque cpt < taille **Faire**

cpt := cpt + 1 // affichage du nouveau nombre de la table

Ecrire ('table [', cpt, ']' = ' , tabent[cpt])

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 14
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

7.2 JEUX D'ESSAI : trier un tableau d'entiers par la méthode de tri par remontée des bulles. (procédure tri_bulles)

Cette procédure reçoit un tableau d'entiers de n éléments, et il faut trier les "taille" premiers entiers de ce tableau (correspondant au nombre d'entiers effectivement dans le tableau).

1) taille = 0

2) taille = 1

3) taille = 10

3.1) valeurs des éléments du tableau: 4 1 3 9 6 8 7 0 5 2

3.2) valeurs des éléments du tableau: 0 4 3 1 2 8 5 6 7 9

3.3) valeurs des éléments du tableau: 2 3 4 5 1 6 7 8 9 0

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 15
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

7.3 SOLUTION : trier un tableau d'entiers par la méthode de tri par remontée des bulles. (procédure tri_bulles 1)

// cette première version du tri à remontée des bulles arrête de trier le tableau
// quand le tableau d'entiers est parcouru sans faire une seule inversion.

Procédure tri_bulles (**Entrée Sortie** tab : table , **Entrée** lgutile : **entier**)

// Cette procédure trie le tableau d'entiers sur sa longueur utile, selon la méthode de remontée des bulles

// tab est : en entrée le tableau d'entiers à trier.
// en sortie le tableau d'entiers triés.
// lgutile est la longueur utile du tableau d'entiers.

Variables

i	:	entier	// indice de parcours de la table d'entiers
inversion	:	booléen	// vrai quand il y a eu au moins une inversion lors // du parcours du tableau d'entiers
tampon	:	entier	// variable intermédiaire permettant l'inversion de // deux entiers du tableau

Début // parcours jusqu'à ne plus avoir d'inversion

Répéter

inversion := **Faux** // pour ce nouveau parcours, il n'y a pas d'inversion
i := 1 // début au nouveau parcours

Tantque i < lgutile **Faire** // arrêt sur le dernier élément du tableau

// comparaison de l'élément à son suivant
Si tab [i] > tab [i + 1] **Alors**
// il faut inverser les deux éléments du tableau
tampon := tab [i]
tab [i] := tab [i + 1]
tab [i + 1] := tampon
// il y a eu au moins une inversion
inversion := **Vrai**

Finsi
i := i + 1 // passage à l'élément suivant

Fintantque

Jusqu'à Non inversion // le tableau est trié quand il n'y a pas eu d'inversion
// lors d'un parcours du tableau

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 16
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

7.4 SOLUTION : trier un tableau d'entiers par la méthode de tri par remontée des bulles. (procédure tri_bulles 2)

// cette deuxième version du tri à remontée des bulles arrête de trier le
// tableau quand le tableau d'entiers est parcouru sans faire une seule
// inversion. De plus, la plus grande valeur tombant au fond à chaque
// parcours, le parcours est diminué d'un élément à chaque cycle.

Procédure tri_bulles (**Entrée Sortie** tab : table , **Entrée** lgutile : **entier**)

// Cette procédure trie le tableau d'entiers sur sa longueur utile, selon la méthode de remontée des bulles

// tab est : en entrée le tableau d'entiers à trier.
// en sortie le tableau d'entiers triés.
// lgutile est la longueur utile du tableau d'entiers.

Variables

i	: entier	// indice de parcours de la table d'entiers
inversion	: booléen	// vrai quand il y a eu au moins une inversion lors // du parcours du tableau d'entiers
tampon	: entier	// variable intermédiaire permettant l'inversion de // deux entiers du tableau

Début // parcours jusqu'à ne plus avoir d'inversion

Répéter

inversion := **Faux** // pour ce nouveau parcours, il n'y a pas d'inversion
i := 1 // début au nouveau parcours

Tantque i < lgutile **Faire** // arrêt sur le dernier élément du tableau

// comparaison de l'élément à son suivant
Si tab [i] > tab [i + 1] **Alors**
// il faut inverser les deux éléments du tableau
tampon := tab [i]
tab [i] := tab [i + 1]
tab [i + 1] := tampon
// il y a eu au moins une inversion
inversion := **Vrai**

Finsi

i := i + 1 // passage à l'élément suivant

Fintantque

lgutile := lgutile - 1 // le plus grand entier est en bas, plus de tri sur lui

Jusqu'à Non inversion // le tableau est trié quand il n'y a pas eu d'inversion
// lors d'un parcours du tableau

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 17
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

7.5 SOLUTION : trier un tableau d'entiers par la méthode de tri par remontée des bulles. (procédure tri_bulles 3)

// cette troisième version du tri à remontée des bulles arrête de trier le
// tableau quand il ne reste plus de nombre à trier. L'emplacement de
// dernière inversion donne la longueur utile pour le prochain parcours.

Procédure tri_bulles (**Entrée Sortie** tab : table , **Entrée** lgutile : **entier**)

// Cette procédure trie le tableau d'entiers sur sa longueur utile, selon la méthode de remontée des bulles

// tab est : en entrée le tableau d'entiers à trier.
// en sortie le tableau d'entiers triés.
// lgutile est la longueur utile du tableau d'entiers.

Variables

i	:	entier	// indice de parcours de la table d'entiers
tampon	:	entier	// variable intermédiaire permettant l'inversion de // deux entiers du tableau
dernier_inver	:	entier	// indique l'endroit de la dernière inversion ou 0

Début // parcours jusqu'à ce qu'il n'y ait plus de nombre à trier

Tantque lgutile > 1 **Faire** // le tableau est trié quand il n'y a plus de nombres à trier

dernier_inver := 0 // pour ce nouveau parcours, il n'y a pas d'inversion

i := 1 // début au nouveau parcours

Répéter // arrêt sur le dernier élément du tableau

// comparaison de l'élément à son suivant

Si tab [i] > tab [i + 1] Alors
// il faut inverser les deux éléments du tableau
tampon := tab [i]
tab [i] := tab [i + 1]
tab [i + 1] := tampon
// noter l'emplacement de la dernière inversion
dernier_inver := i

Finsi

i := i + 1 // passage à l'élément suivant

Jusqu'à i = lgutile // arrêt sur le dernier élément du tableau

lgutile := dernier_inver // le prochain parcours ne triera pas la fin du
// tableau qui est déjà trié

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 18
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

7.6 SOLUTION : trier un tableau d'entiers par la méthode de tri par remontée des bulles.(procédure tri_bulles 4)

```
// cette quatrième version du tri à remontée des bulles arrête de trier le
// tableau quand il ne reste plus de nombre à trier. L'emplacement de la
// dernière inversion donne la longueur utile pour le prochain parcours.
// L'emplacement de la première inversion permet de connaître, à un près,
// le début du prochain parcours.
```

Procédure tri_bulles (Entrée Sortie tab : table , Entrée lgutile : entier)

// Cette procédure trie le tableau d'entiers sur sa longueur utile, selon la méthode de remontée des bulles

```
// tab est : en entrée le tableau d'entiers à trier.
// en sortie le tableau d'entiers triés.
// lgutile est la longueur utile du tableau d'entiers.
```

Variables

```
i : entier // indice de parcours de la table d'entiers
tampon : entier // variable intermédiaire permettant l'inversion de
// deux entiers du tableau
dernier_inver : entier // indique l'endroit de la dernière inversion ou 0
première_inver : entier // indique l'endroit de la première inversion
```

Début // initialisation du début du parcours en début de table

```
première_inver := 1
```

```
// parcours jusqu'à ce qu'il n'y ait plus de nombre à trier
```

Tantque lgutile > 1 **Faire** // le tableau est trié quand il n'y a plus de nombres à trier

```
dernier_inver := 0 // il n'y a pas encore d'inversion pour ce parcours
i := première_inver - 1 // début au nouveau parcours
```

Si i = 0 **Alors** // si la première inversion se fait en début de table,
i := 1 // le prochain parcours se fait à partir du début

Finsi

Répéter // arrêt sur le dernier élément du tableau
// comparaison de l'élément à son suivant

```
Si tab [ i ] > tab [ i + 1 ] Alors
// il faut inverser les deux éléments du tableau
tampon := tab [ i ]
tab [ i ] := tab [ i + 1 ]
tab [ i + 1 ] := tampon
// noter l'emplacement de la première inversion
Si dernier_inver = 0 Alors
premier_inver := i
```

Finsi
// noter l'emplacement de la dernière inversion
dernier_inver := i

Finsi

```
i := i + 1 // passage à l'élément suivant
```

Jusqu'à i = lgutile // arrêt sur le dernier élément du tableau

```
lgutile := dernier_inver // le prochain parcours ne triera pas la fin du
// tableau qui est déjà trié
```

Fintantque

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 19
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

Fin

afpa®	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 20
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

8. EXERCICE : rechercher par dichotomie un élément d'une table classée.

Exemple table des prénoms

1 ->	agathe
2 ->	berthe
3 ->	cellulite
4 ->	cunegonde
5 ->	olga
6 ->	raymonde
7 ->	sidonie

Les prénoms sont classés par ordre alphabétique ;
On connaît la taille de la table (7 ici)

<- M0

<- M2

<- M1

On cherche 'olga' dans la table.

Principe : On partitionne la table en deux sous-tables et un élément médian, et, suivant le résultat de la comparaison de l'élément médian et du prénom recherché (plus grand, plus petit, ou égal) on recommence sur une des deux "sous-table" de la table la recherche, jusqu'à avoir trouvé ou obtenir une sous-table vide (le prénom est alors absent de la table).

chercher('olga') : milieu c'est l'élément n°4

(sur M0)

'olga' > élément n°4

on traite le tableau du dessous (5 à 7)

milieu c'est l'élément n°6

(sur M1)

'olga' < élément n°6

on traite le tableau au dessus (5 à 5)

milieu c'est l'élément n°5

(sur M2)

'olga' = élément n°5

Donnez l'algorithme de la procédure qui donne le numéro de l'élément cherché ou zéro s'il n'y est pas. On suppose que l'on sait comparer des chaînes de caractères.

8.1 SOLUTION : *rechercher par dichotomie un élément d'une table classée.(programme)*

Programme test_dichotomie

// Ce programme met en oeuvre la recherche dichotomique d'un prénom dans un tableau de
// prénoms classés par ordre alphabétique

Constantes n = 80 // nombre maximum de prénoms dans le tableau
 np = 20 // nombre maximum de caractères d'un prénom

Types petitnom = **tableau** [np] **de caractères** // type des prénoms de la table
 table = **tableau** [n] **de** petitnom // type des tables de prénoms que l'on peut traiter

<u>Variables</u>		
tableprénom	: table	// table de prénoms dans laquelle on va chercher
taille	: <u>entier</u>	// longueur utile de la table de prénoms
résultat	: <u>entier</u>	// position du prénom dans la table ou 0 si absent
prénom	: petitnom	// prénom que l'on va chercher dans la table
cpt	: <u>entier</u>	// compteur des prénoms rentrés dans la table

Procédure dichotomie (**Entrée** tabprénom : table, **Entrée** lgutile : **entier**, **Entrée** prénomcherché : petitnom, **Sortie** indice : **entier**)

```
// cette procédure cherche, par dichotomie, un prénom dans un tableau
// de "lgutile" prénoms, classés par ordre alphabétique, et
// retourne l'indice du prénom dans la table ou 0 si il en est absent.
```

```
// tabprénom est le tableau des prénoms.
// lgutile est la longueur utile du tableau de prénoms.
// prénomcherché est le prénom que l'on cherche dans la table.
// indice est la position du prénom dans la table ou 0 si il est absent.
```

Début // saisie de la table de prénoms à traiter et de sa longueur utile

Ecrire ('donnez le nombre de prénoms à entrer dans le tableau : ')

Lire (taille)

cpt := 0

Tantque cpt < taille **Faire**
 cpt := cpt + 1 // saisie d'un nouveau prénom de la table
Ecrire ('donnez le prénom numéro ',cpt,' de la table')
Lire (tableprénom [cpt])

Fintantque

// saisie du prénom à chercher dans le tableau

Ecrire ('donnez le prénom à chercher dans le tableau : ')

Libre (prénom)

// appel de la procédure qui fait tout le travail
dichotomie (tableprénom, taille, prénom, résultat)

```
// résultat de la recherche
```

Si résultat = 0 **Alors**
Ecrire ('Le prénom ', prénom, ' n'est pas dans la table')

Sinon ('Le prénom ' , prénom , ' est pas dans la table ' , résultat)

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 22
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

8.2 JEUX D'ESSAI : rechercher par dichotomie un élément d'une table classée.(procédure dichotomie)

Cette procédure reçoit un tableau de prénoms de n éléments, et il faut chercher le prénom donné dans les "taille" premiers prénoms de ce tableau (correspondant au nombre de prénoms effectivement dans le tableau).

1) *taille* = 0 prénom = **camomille** résultat = 0

2) *taille* = 1 tableprénom = **agathe**

2.1) prénom = **camomille** résultat = 0

2.2) prénom = **agathe** résultat = 1

3) *taille* = 2 tableprénom = **berthe olga**

3.1) prénom = **camomille** résultat = 0

3.2) prénom = **agathe** résultat = 0

3.3) prénom = **sidonie** résultat = 0

3.4) prénom = **berthe** résultat = 1

3.5) prénom = **olga** résultat = 2

4) *taille* = 7 tableprénom = **agathe berthe cellulite cunégonde olga raymonde sidonie**

4.1) prénom = **camomille** résultat = 0

4.2) prénom = **cunégonde** résultat = 4

4.3) prénom = **olga** résultat = 5

4.4) prénom = **agathe** résultat = 1

4.5) prénom = **sidonie** résultat = 7

8.3 SOLUTION : *rechercher par dichotomie un élément d'une table classée.(procédure dichotomie)*

Procédure dichotomie (**Entrée** tabprénom : table, **Entrée** lgutile : **entier**,
Entrée prénomcherché : prénom, **Sortie** indice : **entier**)

// cette procédure cherche, par dichotomie, un prénom dans un tableau de
 // "lgutile" prénoms, classés par ordre alphabétique, et retourne l'indice
 // du prénom dans la table ou 0 si il en est absent.

// tabprénom est le tableau des prénoms.
 // lgutile est la longueur utile du tableau de prénoms.
 // prénomcherché est le prénom que l'on cherche dans la table.
 // indice est la position du prénom dans la table ou 0 si il est absent.

Variables idébut : **entier** // indice de début de la zone de recherche
 ifin : **entier** // indice de fin de la zone de recherche

Début // initialisation des indices de zone et médian

idébut := 1
 ifin := lgutile
 indice := (idébut + ifin) **div** 2

// recherche du prénom jusqu'à avoir trouvé, ou que la zone soit vide

Tantque (idébut < ifin) **et** (tabprénom [indice] <> prénomcherché) **Faire**
 // arrêt quand la zone de recherche contient au plus un prénom, ou quand il est trouvé

Si tabprénom [indice] > prénomcherché **Alors**
 ifin := indice - 1 // zone du haut

Sinon
 idébut := indice + 1 // zone du bas

Finsi
 indice := (idébut + ifin) **div** 2

Fintantque

Si (idébut > ifin) **ou** (tabprénom [indice] <> prénom) **Alors**
 // zone de recherche est vide ou prénom absent
 indice := 0

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 24
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

9. EXERCICE : donner la valeur du x-ième bit d'un entier.

Donnez l'algorithme d'une procédure qui, pour un entier, donne la valeur du x-ième bit de cet entier. Cette valeur sera donnée sous la forme d'un booléen : vrai si la valeur est 1, faux si elle est 0.

Le LSB (Lower Significant Bit) a pour numéro 0.

Il existe deux opérateurs sur les entiers qui sont MOD et DIV utilisables de la manière suivante:

7 MOD 3 = 1 reste de la division entière de 7 par 3
7 DIV 3 = 2 résultat de la division entière de 7 par 3.

9.1 JEUX D'ESSAI : *donner la valeur du x-ième bit d'un entier.(procédure valeur_bit)*

<i>nombre</i>	<i>poids</i>	<i>résultat</i>
0	0	faux
1	0	vrai
63000	1	faux
47	4	faux
47	5	vrai
3	-1	faux

9.2 SOLUTION : donner la valeur du x-ième bit d'un entier.(programme)

Programme valeur_bit_de_poids_x
 // Ce programme donne la valeur du bit de poids x d'un entier

Variables

nombre	:	<u>entier</u>	// nombre donné par l'opérateur
rang	:	<u>entier</u>	// rang du bit dans le nombre (0 = premier bit)
résultat	:	<u>booléen</u>	// vrai quand le bit n°rang est à 1 dans nombre

Procédure valeur_bit (Entrée valnb : entier , Entrée poidsbit : entier , Sortie valbit : booléen)

// Calcul de la valeur du bit de poids x d'un nombre.
 // Le bit numéro zéro est le bit de poids le plus faible.
 // Si le poids du bit est négatif, la procédure rendra valbit à faux.

// valnb est le nombre dont on veut connaître la valeur d'un bit
 // poidsbit est le poids du bit concerné (0 pour le bit de poids faible)
 // valbit représente la valeur du bit (vrai si le bit est à 1 dans valnb)

Début // saisie du nombre et du rang du bit à traiter

Ecrire ('donnez la valeur du nombre')

Lire (nombre)

Ecrire ('donnez le rang du bit dont vous voulez la valeur (0 = premier bit) ')

Lire (rang)

// appel de la procédure qui fait tout le travail

valeur_bit (nombre, rang, résultat)

// résultat de la recherche

Si résultat **Alors**
Ecrire ('La valeur du bit de poids 'rang,' dans le nombre 'nombre,' est 1')

Sinon
Ecrire ('La valeur du bit de poids 'rang,' dans le nombre 'nombre,' est 0')

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 26
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

9.3 SOLUTION : donner la valeur du x-ième bit d'un entier.(procédure valeur_bit)

Procédure valeur_bit (Entrée valnb : entier , Entrée poidsbit : entier ,
Sortie valbit : booléen)

// Calcul de la valeur du bit de poids x d'un nombre.
// Le bit numéro zéro est le bit de poids le plus faible.
// Si le poids du bit est négatif, la procédure rendra valbit à faux.

// valnb est le nombre dont on veut connaître la valeur d'un bit
// poidsbit est le poids du bit concerné (0 pour le bit de poids faible)
// valbit représente la valeur du bit (vrai si le bit est à un dans valnb)

Début

Si poidsbit < 0 Alors
valbit := Faux // il faut mettre un paramètre de plus

Sinon
// divisions successives jusqu'à avoir un poids ou un nombre nul

Tantque (poidsbit > 0) et (valnb <> 0) Faire

valnb := valnb div 2
poidsbit := poidsbit - 1

Fintantque

// récupération du LSB de valnb

valbit := ((valnb mod 2) = 1)

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 27
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

10. EXERCICE : faire le ET logique de deux entiers.

Donnez l'algorithme d'une procédure qui fait un ET logique bit à bit entre un nombre entier donné et un masque donné. Le résultat sera donné sous forme d'un nouvel entier.

Exemple : ici, a, b,..., h sont des valeurs de bit à 1 ou 0.

valeur en entrée :	a	b	c	d	e	f	g	h
valeur du masque:	1	0	1	0	0	1	1	0
Résultat:	a	0	c	0	0	f	g	0

10.1 SOLUTION : faire le ET logique de deux entiers dans un troisième.(programme)**Programme** et_logique_de_deux_entiers

// Ce programme fait le et logique de deux entiers dans un troisième

Variables

nombre	:	<u>entier</u>	// nombre donné par l'opérateur
masque	:	<u>entier</u>	// masque avec lequel on fait le et logique
résultat	:	<u>entier</u>	// résultat du et logique entre masque et nombre

Procédure et_logique (Entrée valnb : entier , Entrée valmask : entier , Sortie valres : entier)

// Cette procédure calcule le ET logique entre un nombre et un masque.
 // valnb est le nombre dont on veut faire le ET logique
 // valmask est le masque avec lequel on fait le ET logique
 // valres est le résultat du ET logique entre valnb et valmask

Début // saisie du nombre et du masque

Ecrire ('donnez la valeur du nombre')

Lire (nombre)

Ecrire ('donnez la valeur du masque')

Lire (masque)

// appel de la procédure qui fait tout le travail

et_logique (nombre, masque, résultat)

// affichage du résultat du calcul

Ecrire ('La valeur du et logique entre 'nombre,' et 'masque,' est ',résultat')

Fin

10.2 JEUX D'ESSAI : faire le ET logique de deux entiers dans un troisième.(procédure et_logique)

Remarque : nombre et masque ont un rôle symétrique.

nombre	masque	résultat
25327	0	0
0	25327	0
25327	25327	25327
14	15	14
15	14	14
15	16	0
16	15	0
65535	65536	0
65536	65535	0

10.3 SOLUTION : faire le ET logique de deux entiers dans un troisième. (procédure et_logique)

Procédure et_logique (Entrée valnb : entier , Entrée valmask : entier , Sortie valres : entier)

// Cet procédure calcule le et logique entre un nombre ET un masque.
 // valnb est le nombre dont on veut faire le ET logique
 // valmask est le masque avec lequel on fait le et logique
 // valres est le résultat du ET logique entre valnb et valmask

variables poids : entier // poids de chaque bit (soit les puissances de 2)

Début // initialisation du poids et du résultat

valres := 0
 poids := 1

// division successives des deux nombres par deux

Tantque (valmask <> 0) **et** (valnb <> 0) **Faire**
 // arrêt quand le nombre ou le masque est nul

valres := (valmask mod 2) * (valnb mod 2) * poids + valres
 // quand les deux LSB sont à 1 on ajoute le poids

valmask := valmask div 2
 // passage au bit suivant sur les deux entiers

valnb := valnb div 2
 poids := poids * 2 // passage au poids du bit suivant

Fintantque

Fin

11. EXERCICE : coder une phrase en une autre en code ascii.

On désire enregistrer des informations en caractères ASCII étendu sur des étiquettes magnétiques. Hélas le protocole de dialogue avec le programmeur d'étiquettes n'autorise pas d'envoyer de tels caractères. Il n'accepte de transmettre que des caractères imprimables du code ASCII (<80 hexa). Pour pouvoir imprimer correctement les étiquettes, il faut opérer de la sorte :

pour transmettre 'A' (code ascii 41 hexa)

- Envoyer les caractères '4' et '1' (code ascii 34 et 31 hexa)
- Le programmeur d'étiquettes reçoit '4' et '1', associe les 2 caractères et imprime 'A'.

Donnez l'algorithme de la procédure qui, pour une chaîne de caractères de longueur donnée, constitue la chaîne de caractères à envoyer au programmeur d'étiquettes.

afpa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 30
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

11.1 SOLUTION : *coder une phrase en une autre en code ascii. (programme)*

<u>Programme</u>	<code>coder_une_phrase</code>
-------------------------	-------------------------------

```
// Ce programme code une chaîne de caractères en une autre chaîne de caractères
// contenant les caractères composant les codes ascii des caractères de la première chaîne.
// La chaîne de caractères est délimitée par sa longueur calculée.
```

```
Constantes    taille    = 80      // nombre maximum de caractères dans la chaîne
                  carfin    = '.'      // caractère de terminaison de la phrase
```

Types

chaîne = **tableau** [taille] **de caractères**
// type des chaînes de caractères traitées

chaîncodée = **tableau** [taille * 2] **de caractères**
// type des chaînes de caractères codées

<u>Variables</u>		
phrase	:	chaîne // phrase dans laquelle on va compter les carcibles
longueur	:	<u>entier</u> // longueur de la phrase donnée par l'opérateur
résultat	:	chaîncodée // chaîne de caractères codée pour la transmission

Procédure coder_ascii (**Entrée** texte : chaîne , **Entrée** taille : entier , **Sortie** code : chaînecodée)

```
// Cette procédure code une chaîne en une autre chaîne contenant les
// caractères composant les codes ascii de la première chaîne.
// La chaîne résultat est donc deux fois plus longue que celle initiale.
// Elle ne contient que des caractères '0' ... '9', et 'A' ... 'F'.
```

```
// texte est la chaîne de caractères à traiter.  
// taille est la longueur de la chaîne à traiter.  
// code est la chaîne codée prête pour la transmission.
```

Début // saisie de la phrase à traiter et de sa longueur

Ecrire ('donnez la phrase à coder ')
Lire (phrase)
Ecrire ('donnez la longueur de la phrase')
Lire (longueur)

`coder_ascii (phrase, longueur, résultat)`

```
// résultat du codage
```

Ecrire ('la phrase codée est ' , résultat)

Ecrire ('la longueur de la phrase codée est ' , 2 * longueur)

Fin

11.2 JEUX D'ESSAI : coder une phrase en une autre en code ascii.(procédure coder_ascii)

taille	texte	code
0	~	~
1	'A'	'41'
5	'ABC32'	'4142433332'

11.3 SOLUTION : coder une phrase en une autre en code ascii.(procédure coder_ascii)

Procédure coder_ascii (Entrée texte : chaîne , Entrée taille : entier , Sortie code : chaînecodée)

// Cette procédure code une chaîne en une autre chaîne contenant les
 // caractères composant les codes ascii de la première chaîne.
 // La chaîne résultat est donc deux fois plus longue que celle initiale.
 // Elle ne contient que des caractères '0' ... '9', et 'A' ... 'F'.

// texte est la chaîne de caractères à traiter.
 // taille est la longueur de la chaîne à traiter.
 // code est la chaîne codée prête pour la transmission.

Variables i : entier // indice de parcours du texte à coder
 valeur : entier // code ascii du caractère en cours de traitement

Fonction ordre (Entrée élément : caractère) : entier

// Cette fonction permet de connaître le code ascii d'un caractère.

// élément est le caractère dont on veut connaître le code ascii.
 // ordre retourne le code ascii du caractère concerné.

Fonction carhexa (Entrée nombre : entier) : caractère

// Cette fonction retourne le caractère hexadécimal permettant de coder
 // un entier compris entre 0 et 15. En cas d'erreur sur la valeur de l'entier
 // la valeur retournée est le caractère '#'.
 // nombre est un entier entre 0 et 15 .
 // carhexa retourne le chiffre hexadécimal correspondant au nombre
 // ou le caractère '#' si il y a une erreur sur le nombre.

Début // initialisation de l'indice de parcours

i := 1

// parcours du texte à coder, en codant chaque caractère

Tantque i <= taille **Faire**

// arrêt quand tous les caractères sont codés
 valeur := ordre (texte [i])

// codage du chiffre hexa de poids fort

code [2 * i - 1] := carhexa (valeur div 16)

// codage du chiffre hexa de poids faible

code [2 * i] := carhexa (valeur mod 16)

i := i + 1

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 32
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

11.4 SOLUTION : coder une phrase en une autre en code ascii.(fonction ordre)

Fonction ordre (**Entrée** élément : **caractère**) : **entier**

// Cette fonction permet de connaître le code ascii d'un caractère.

// élément est le caractère dont on veut connaître le code ascii.

// ordre retourne le code ascii du caractère concerné.

Cette fonction est un outil de quasiment tous les environnements de programmation. Il est donc inutile de vouloir le recréer.

11.5 JEUX D'ESSAI : coder une phrase en une autre en code ascii.(fonction carhexa)

nombre

0 à 15

-5

16

45

carhexa

'0' à '9', 'A' à 'F'

#'

#'

#'

11.6 SOLUTION : coder une phrase en une autre en code ascii.(fonction carhexa)

Fonction carhexa (Entrée nombre : entier) : caractère

// Cette fonction retourne le caractère hexadécimal permettant de coder
 // un entier compris entre 0 et 15. En cas d'erreur sur la valeur de l'entier
 // la valeur retournée est le caractère '#'.

// nombre est un entier entre 0 et 15 .
 // carhexa retourne le chiffre hexadécimal correspondant au nombre
 // ou le caractère '#' si il y a une erreur sur le nombre.

Constantes erreur = '#' // caractère retourné si le nombre est erroné.

Variables hexa : caractère // chiffre hexadécimal représentant le nombre.

Début // suivant la valeur du nombre, renvoi du caractère adéquat

choix sur nombre Faire

0	:	hexa := '0'
1	:	hexa := '1'
2	:	hexa := '2'
3	:	hexa := '3'
4	:	hexa := '4'
5	:	hexa := '5'
6	:	hexa := '6'
7	:	hexa := '7'
8	:	hexa := '8'
9	:	hexa := '9'
10	:	hexa := 'A'
11	:	hexa := 'B'
12	:	hexa := 'C'
13	:	hexa := 'D'
14	:	hexa := 'E'
15	:	hexa := 'F'
<u>Autrecas</u>	:	hexa := erreur

Finchoix

retourner (hexa) // résultat de la fonction

Fin

12. EXERCICE : chercher un mot dans une phrase.

Soit une phrase terminée par un point. Donnez l'algorithme de la procédure qui pour un mot donné (mot et longueur du mot) détermine si le mot est dans la phrase.

12.1 SOLUTION : chercher un mot dans une phrase.(algorithme de principe)

Répéter

prendre un mot

Si le mot a la même longueur que le mot cherché **Alors**
comparer les deux mots

Finsi

Jusqu'à avoir trouvé le mot **ou** avoir parcouru toute la phrase

Définition des données :

le mot cherché est donné par :

une chaîne de caractères de type chaînemot
 sa longueur de type entier

un mot de la phrase est donné par :

la phrase initiale
 l'indice juste après le mot ou sur le caractère terminateur
 la longueur du mot ou 0 si le mot est le caractère terminateur

La phrase est donnée par :

une chaîne de caractères de type chaîne
 un caractère terminateur carterm

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 35
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

12.2 SOLUTION : chercher un mot dans une phrase.(interfaces)

Procédure chercher_mot (Entrée texte : chaîne , Entrée mot_cherché : chaînemot,
Entrée long : entier, Sortie trouvé : booléen)

// Cette procédure cherche un mot dans une chaîne de caractères.
 // Si le mot se trouve dans la chaîne, alors le résultat est vrai.

// texte est la chaîne de caractères à traiter
 // mot_cherché est le mot recherché dans la chaîne
 // long est la longueur du mot à chercher
 // trouvé est vrai si le mot cherché est dans la chaîne

Procédure prendre_mot (Entrée texte : chaîne, Entrée Sortie ind : entier, Sortie lg : entier)

// Cette procédure positionne un indice sur le caractère suivant le mot
 // repéré, et donne sa longueur, ou positionne l'indice sur le caractère
 // terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
 // ind est en entrée la position à partir de laquelle on cherche un mot
 // en sortie la position juste après le mot ou carterm
 // lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure comparer_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
Entrée mot : chaînemot, Sortie égal : booléen)

// Cette procédure compare un mot repéré dans la chaîne par sa position
 // "après", à un mot donné. Les deux mots ont même longueur

// texte est la chaîne où se trouve le mot
 // ind est la position dans texte juste après le mot
 // longmot est la longueur des deux mots
 // mot est le mot recherché
 // égal est vrai si les deux mots sont égaux

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 36
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

12.3 SOLUTION : chercher un mot dans une phrase.(programme)

Programme chercher_mot_phrase

// Ce programme cherche si un mot se trouve dans une phrase. Par mot, on entend une
 // suite de caractères délimités par le début ou la fin de la phrase ou des espaces.

Constantes taille = 80 // nombre maximum de caractères dans la chaîne
 max = 20 // nombre maximum des caractères d'un mot
 carterm = '.' // caractère terminant la phrase à traiter

Types chaîne = tableau [taille] de caractères // type des chaînes de caractères traitées
 chaînemot = tableau [max] de caractères // type du mot recherché dans la phrase

Variables phrase : chaîne // phrase est la chaîne où l'on cherche le mot
 mot : chaînemot // mot recherché dans la phrase
 longueur : entier // longueur du mot recherché
 résultat : booléen // vrai si le mot est dans la phrase

Procédure chercher_mot (Entrée texte : chaîne , Entrée mot_cherché : chaînemot,
 Entrée long : entier, Sortie trouvé : booléen)

// Cette procédure cherche un mot dans une chaîne de caractères.
 // Si le mot se trouve dans la chaîne, alors le résultat est vrai.

// texte est la chaîne de caractères à traiter
 // mot_cherché est le mot recherché dans la chaîne
 // long est la longueur du mot à chercher
 // trouvé est vrai si le mot cherché est dans la chaîne

Début // saisie de la phrase à traiter

Ecrire ('donnez la phrase terminée par un caractère 'carterm, ' : ')

Lire (phrase)

Ecrire ('donnez le mot recherché : ') // saisie du mot et de sa longueur

Lire (mot)

Ecrire ('donnez la longueur du mot : ')

Lire (longueur)

chercher_mot (phrase, mot, longueur, résultat)

Si résultat Alors // résultat de la recherche

Ecrire ('le mot ' , mot, ' est dans la phrase. ')

Sinon

Ecrire ('le mot ' , mot, ' n'est pas dans la phrase. ')

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 37
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

12.4 JEUX D'ESSAI : chercher un mot dans une phrase.(procédure chercher_mot)

- 1) phrase = '.'
mot = ''
longueur = 0
résultat = **vrai**
- 2) phrase = 'le chat est gris.'
mot = ''
longueur = 0
résultat = **vrai**
- 3) phrase = '.'
mot = 'chat'
longueur = 4
résultat = **faux**
- 4) phrase = 'le chat est gris.'
mot = 'chat'
longueur = 4
résultat = **vrai**
- 5) phrase = 'le chat est gris.'
mot = 'cheri'
longueur = 5
résultat = **faux**
- 6) phrase = ' chat est gris.'
mot = 'chat'
longueur = 4
résultat = **vrai**
- 7) phrase = 'chat est gris.'
mot = 'chat'
longueur = 4
résultat = **vrai**
- 8) phrase = 'le chat .'
mot = 'chat'
longueur = 4
résultat = **vrai**
- 9) phrase = 'le chat.'
mot = 'chat'
longueur = 4
résultat = **vrai**
- 10) phrase = 'le chat.'
mot = 'chat.'
longueur = 5
résultat = **faux**
// le caractère terminateur ne fait pas partie de la phrase
- 11) phrase = 'le chaton est gris.'
mot = 'chat'
longueur = 4
résultat = **faux**
- 12) phrase = 'un achat .'
mot = 'chat'
longueur = 4
résultat = **faux**

12.5 SOLUTION : chercher un mot dans une phrase.(procédure chercher_mot)

Procédure

chercher_mot (**Entrée** texte : chaîne , **Entrée** mot_cherché : chaînemot,
Entrée long : **entier**, **Sortie** trouvé : **booléen**)

// Cette procédure cherche un mot dans une chaîne de caractères.

// Si le mot se trouve dans la chaîne, alors le résultat est vrai.

// texte est la chaîne de caractères à traiter

// mot_cherché est le mot recherché dans la chaîne

// long est la longueur du mot à chercher

// trouvé est vrai si le mot cherché est dans la chaîne

// carterm est une constante définie en amont de la procédure

// carterm est le caractère terminateur des chaînes

Variables

i : **entier** // indice de parcours de la chaîne

taille_mot : **entier** // longueur d'un mot repéré dans le texte

Procédure

prendre_mot (**Entrée** texte : chaîne, **Entrée Sortie** ind : **entier**, **Sortie** lg : **entier**)

// Cette procédure positionne un indice sur le caractère suivant le mot

// repéré, et donne sa longueur, ou positionne l'indice sur le caractère

// terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot

// ind est en entrée la position à partir de laquelle on cherche un mot

// en sortie la position juste après le mot ou carterm

// lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure

comparer_mot (**Entrée** texte : chaîne , **Entrée** ind : **entier**, **Entrée** longmot : **entier**,
Entrée mot : chaînemot, **Sortie** égal : **booléen**)

// Cette procédure compare un mot repéré dans la chaîne par sa position

// "après", à un mot donné. Les deux mots ont même longueur

// texte est la chaîne où se trouve le mot

// ind est la position dans texte juste après le mot

// longmot est la longueur des deux mots

// mot est le mot recherché

// égal est vrai si les deux mots sont égaux

Début

// indice au début de la chaîne et initialisation de trouvé

i := 1

trouvé := **faux**

// parcours de la phrase à la recherche des lettres du mot

Répéter

prendre_mot (texte, i, taille_mot) // repérage d'un mot dans le texte

Si taille_mot = long **Alors** // si la longueur correcte, comparaison des mots
comparer_mot (texte, i, long, mot_cherché, trouvé)

Finsi

Jusqu'à trouvé **ou** (taille_mot = 0) // arrêt quand le mot est trouvé, ou quand terminateur trouvé

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 39
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

Fin**12.6 JEUX D'ESSAI : chercher un mot dans une phrase (procédure prendre_mot).**

Le caractère espace sera ici représenté par le caractère **␣**.

- | | |
|---|--|
| 1) <i>texte</i> = '.'
<i>ind</i> = 1 | <i>longueur</i> = 0
<i>ind</i> = 1 |
| 2) <i>texte</i> = '␣␣␣.'
<i>ind</i> = 1 | <i>longueur</i> = 0
<i>ind</i> = 4 |
| 3) <i>texte</i> = 'le␣chat␣est␣gris.'
<i>ind</i> = 1 | <i>longueur</i> = 2
<i>ind</i> = 3 |
| 4) <i>texte</i> = '␣␣␣le␣chat.'
<i>ind</i> = 1 | <i>longueur</i> = 2
<i>ind</i> = 6 |
| 5) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 3 | <i>longueur</i> = 4
<i>ind</i> = 9 |
| 6) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 13 | <i>longueur</i> = 4
<i>ind</i> = 18 |
| 7) <i>texte</i> = 'le␣␣chat␣est␣gris␣␣.'
<i>ind</i> = 13 | <i>longueur</i> = 4
<i>ind</i> = 18 |
| 8) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 18 | <i>longueur</i> = 0
<i>ind</i> = 18 |
| 9) <i>texte</i> = 'le␣␣chat␣est␣gris␣␣.'
<i>ind</i> = 18 | <i>longueur</i> = 0
<i>ind</i> = 20 |

12.7 SOLUTION : chercher un mot dans une phrase.(procédure prendre_mot)

Procédure prendre_mot (**Entrée** texte : chaîne, **Entrée Sortie** ind : **entier**, **Sortie** lg : **entier**)

// Cette procédure positionne un indice sur le caractère suivant le mot
 // repéré, et donne sa longueur, ou positionne l'indice sur le caractère
 // terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
 // ind est en entrée la position à partir de laquelle on cherche un mot
 // en sortie la position juste après le mot ou carterm
 // lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

// carterm est une constante définie en amont de la procédure
 // carterm est le caractère terminateur des chaînes

Constantes espace = ' ' // caractère séparateur de mots

Début // éventuels espaces en début de mot sautés

Tantque texte [ind] = espace **Faire** // arrêt quand on a autre chose qu'un espace

ind := ind + 1

Fintantque

// initialisation de la longueur

lg := 0

// recherche de la fin du mot

Tantque (texte [ind] <> espace) **et** (texte [ind] <> carterm) **Faire**
 // arrêt quand on a un espace ou le carterm

ind := ind + 1

lg := lg + 1

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 41
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

12.8 JEUX D'ESSAI : chercher un mot dans une phrase (procédure comparer_mot).

Le caractère espace sera remplacé ici par le caractère α .

- 1) texte = 'titi.'
ind = 5
longmot = 4
mot = 'toto' égal = faux
- 2) texte = 'le α chat α .'
ind = 8
longmot = 4
mot = 'toto' égal = faux
- 3) texte = 'polo.'
ind = 5
longmot = 4
mot = 'toto' égal = faux
- 4) texte = 'tout.'
ind = 5
longmot = 4
mot = 'toto' égal = faux
- 5) texte = 'toto α est α ici.'
ind = 5
longmot = 4
mot = 'toto' égal = vrai
- 6) texte = 'la α tête α à α toto.'
ind = 15
longmot = 4
mot = 'toto' égal = vrai
- 7) texte = 'le α chat α est α gris.'
ind = 8
longmot = 4
mot = 'chat' égal = vrai
- 8) texte = 'le α chat α est α gris.'
ind = 17
longmot = 0
mot = '' égal = vrai

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 42
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

12.9 SOLUTION : chercher un mot dans une phrase.(procédure comparer_mot)

Procédure comparer_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
Entrée mot : chaînemot, Sortie égal : booléen)

// Cette procédure compare un mot repéré dans la chaîne par sa position
// "après", à un mot donné. Les deux mots ont même longueur

// texte est la chaîne où se trouve le mot
// ind est la position dans texte juste après le mot
// longmot est la longueur des deux mots
// mot est le mot recherché
// égal est vrai si les deux mots sont égaux

Début // indice sur le dernier caractère du mot de la phrase

ind := ind - 1

// parcours du mot en comparant avec le mot de la phrase

Tantque (longmot <> 0) **et** (texte [ind] = mot [longmot]) **Faire**
// arrêt quand le mot est parcouru, ou quand les deux mots sont différents
ind := ind - 1
longmot := longmot - 1

Fintantque

// résultat de la comparaison

égal := (longmot = 0)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 43
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

13. EXERCICE : recopier une phrase en inversant chaque mot.

Soit une phrase terminée par un point.

Donnez l'algorithme de la procédure qui inverse chacun des mots de cette phrase et qui rend le résultat dans une deuxième phrase. Cette deuxième phrase comportera un espace entre deux mots et le point final sera cadré le plus à gauche possible.

Exemple :

le chat est gris.

donnera

el tahc tse sirg.

(un seul espace entre les mots)

13.1 SOLUTION : *recopier une phrase en inversant chaque mot. (algorithme de principe)*

prendre un mot

Tantque la longueur du mot est non nulle **Faire**

recopier le mot en l'inversant

prendre un mot

Si le mot a une longueur non nulle **Alors**

mettre un espace

Finsi

Fintantque

mettre le point final

Définition des données:

un mot de la phrase est donné par :

la phrase initiale

l'indice juste après le mot ou sur le caractère terminateur

la longueur du mot ou 0 si le mot est le caractère terminateur

La phrase est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm

La phrase résultat est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 44
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

13.2 SOLUTION : *recopier une phrase en inversant chaque mot.(interfaces)*

Procédure recopier_inverser (Entrée texte : chaîne , Sortie inverse : chaîne)

// Cette procédure recopie le texte en inversant chaque mot.
 // Dans la chaîne résultat il y aura le minimum d'espaces.

// texte est la chaîne de caractères à traiter
 // inverse est la phrase obtenue en inversant chaque mot

Procédure prendre_mot (Entrée texte : chaîne, Entrée Sortie ind : entier, Sortie lg : entier)

// Cette procédure positionne un indice sur le caractère suivant le mot
 // repéré, et donne sa longueur, ou positionne l'indice sur le caractère
 // terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
 // ind est en entrée la position à partir de laquelle on cherche un mot
 // en sortie la position juste après le mot ou carterm
 // lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure inverser_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
 Entrée Sortie indinv : entier, Entrée Sortie inverse : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position
 // " après ", en inversant les lettres du mot, à l'emplacement indiqué
 // par indinv dans la chaîne inverse

// texte est la chaîne où se trouve le mot
 // ind est la position dans texte juste après le mot
 // longmot est la longueur du mot
 // indinv est en entrée l'endroit où recopier le mot inversé
 // en sortie l'endroit où mettre l'espace ou le point final
 // inverse est en entrée la portion de chaîne qui a déjà été inversée
 // en sortie le mot est recopié à la suite

13.3 SOLUTION : *recopier une phrase en inversant chaque mot.(programme)***Programme** inverser_phrase

// Ce programme recopie une phrase dans une autre en inversant les lettres de chaque
 // mot, et en recopiant le minimum d'espaces dans la phrase résultat.

Constantes taille = 80 // nombre maximum de caractères dans la chaîne
 carterm = '.' // caractère terminant la phrase à traiter

Types chaîne = **tableau** [taille] **de caractères** // type des chaînes de caractères traitées

Variables phrase : chaîne // phrase est la chaîne où l'on cherche le mot
 phrase_résultat : chaîne // phrase_résultat est la phrase inversée

Procédure recopier_inverser (**Entrée** texte : chaîne , **Sortie** inverse : chaîne)

// Cette procédure recopie le texte en inversant chaque mot.
 // Dans la chaîne résultat il y aura le minimum d'espaces.

// texte est la chaîne de caractères à traiter
 // inverse est la phrase obtenue en inversant chaque mot

Début // saisie de la phrase à traiter

Ecrire ('donnez la phrase terminée par un caractère ', carterm, ' : ')

Lire (phrase)

// appel de la procédure qui fait tout le travail

recopier_inverser (phrase, phrase_résultat)

// résultat du travail

Ecrire ('la phrase est : ', phrase)

Ecrire ('la phrase inversée est : ', phrase_résultat)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 46
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

13.4 JEUX D'ESSAI : recopier une phrase en inversant chaque mot.(procédure recopier_inverser)

Le caractère espace est représenté ici par le caractère **■**.

- | | |
|--------------------------------------|---------------------------------------|
| 1) phrase = '.' | phrase_résultat = '.' |
| 2) phrase = '■■■.' | phrase_résultat = '.' |
| 3) phrase = 'le■chat■est■gris.' | phrase_résultat = 'el■tahc■tse■sirr.' |
| 4) phrase = '■■le■■chat■est■■gris■.' | phrase_résultat = 'el■tahc■tse■sirr.' |
| 5) phrase = 'trop.' | phrase_résultat = 'port.' |
| 6) phrase = '■■■trop.' | phrase_résultat = 'port.' |
| 7) phrase = 'trop■■■.' | phrase_résultat = 'port.' |

SOLUTION : recopier une phrase en inversant chaque mot.(procédure recopier_inverser)

Procédure recopier_inverser (**Entrée** texte : chaîne , **Sortie** inverse : chaîne)

// Cette procédure recopie le texte en inversant chaque mot.
// Dans la chaîne résultat il y aura le minimum d'espaces.

// texte est la chaîne de caractères à traiter
// inverse est la phrase obtenue en inversant chaque mot

Constantes espace = ' ' // caractère séparateur de mots

Variables i_t : **entier** // indice de parcours de la chaîne texte
i_i : **entier** // indice de parcours de la chaîne inverse
taille_mot : **entier** // longueur d'un mot repéré dans le texte

Procédure prendre_mot (**Entrée** texte : chaîne, **Entrée Sortie** ind : **entier**, **Sortie** lg : **entier**)

// Cette procédure positionne un indice sur le caractère suivant le mot
// repéré, et donne sa longueur, ou positionne l'indice sur le caractère
// terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
// ind est en entrée la position à partir de laquelle on cherche un mot
// en sortie la position juste après le mot ou carterm
// lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure inverser_mot (**Entrée** texte : chaîne , **Entrée** ind : **entier**, **Entrée** longmot : **entier**,
Entrée Sortie indinv : **entier**, **Entrée Sortie** inverse : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position
// " après ", en inversant les lettres du mot, à l'emplacement indiqué
// par indinv dans la chaîne inverse

// texte est la chaîne où se trouve le mot
// ind est la position dans texte juste après le mot
// longmot est la longueur du mot
// indinv est en entrée l'endroit où recopier le mot inversé
// en sortie l'endroit où mettre l'espace ou le point final
// inverse est en entrée la portion de chaîne qui a déjà été inversée
// en sortie le mot est recopié à la suite

Début // indices au début des chaînes

i_t := 1

i_i := 1

prendre_mot (texte, i_t, taille_mot) // parcours de la phrase mot à mot pour les recopier inversés

Tantque taille_mot <> 0 **Faire** // arrêt quand le terminateur a été détecté

inverser_mot (texte, i_t, taille_mot, i_i, inverse) // recopie du mot en l'inversant

prendre_mot (texte, i_t, taille_mot) // repérage d'un mot dans le texte

Si taille_mot <> 0 **Alors** // si ce n'est pas le terminateur, recopie d'un espace

inverse [i_i] := espace

i_i := i_i + 1

Finsi

Fintantque

inverse [i_i] := carterm // terminateur mis dans la phrase résultat

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 48
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

13.5 JEUX D'ESSAI : recopier une phrase en inversant chaque mot.(procédure_prendre_mot)

Le caractère espace sera ici représenté par le caractère **■**.

- | | |
|---|--|
| 1) <i>texte = '.'</i>
<i>ind = 1</i> | <i>longueur = 0</i>
<i>ind = 1</i> |
| 2) <i>texte = '■■■.'</i>
<i>ind = 1</i> | <i>longueur = 0</i>
<i>ind = 4</i> |
| 3) <i>texte = 'le■chat■est■gris.'</i>
<i>ind = 1</i> | <i>longueur = 2</i>
<i>ind = 3</i> |
| 4) <i>texte = '■■■le■chat.'</i>
<i>ind = 1</i> | <i>longueur = 2</i>
<i>ind = 6</i> |
| 5) <i>texte = 'le■■■chat■est■gris.'</i>
<i>ind = 3</i> | <i>longueur = 4</i>
<i>ind = 9</i> |
| 6) <i>texte = 'le■■■chat■est■gris.'</i>
<i>ind = 13</i> | <i>longueur = 4</i>
<i>ind = 18</i> |
| 7) <i>texte = 'le■■■chat■est■gris■■■.'</i>
<i>ind = 13</i> | <i>longueur = 4</i>
<i>ind = 18</i> |
| 8) <i>texte = 'le■■■chat■est■gris.'</i>
<i>ind = 18</i> | <i>longueur = 0</i>
<i>ind = 18</i> |
| 9) <i>texte = 'le■■■chat■est■gris■■■.'</i>
<i>ind = 18</i> | <i>longueur = 0</i>
<i>ind = 20</i> |

13.6 SOLUTION : *recopier une phrase en inversant chaque mot.(procédure prendre_mot)*

Procédure prendre_mot (**Entrée** texte : chaîne, **Entrée Sortie** ind : **entier**, **Sortie** lg : **entier**)

// Cette procédure positionne un indice sur le caractère suivant le mot
 // repéré, et donne sa longueur, ou positionne l'indice sur le caractère
 // terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
 // ind est en entrée la position à partir de laquelle on cherche un mot
 // en sortie la position juste après le mot ou carterm
 // lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

// carterm et espace sont des constantes définies en amont
 // carterm est le caractère terminateur des chaînes
 // espace est le caractère séparateur de mots

Début // éventuels espaces en début de mot ignorés

Tantque texte [ind] = espace **Faire** // arrêt quand on a autre chose qu'un espace

ind := ind + 1

Fintantque

// initialisation de la longueur

lg := 0

// recherche de la fin du mot

Tantque (texte [ind] <> espace) **et** (texte [ind] <> carterm) **Faire**
 // arrêt quand on a un espace ou le carterm

ind := ind + 1
 lg := lg + 1

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 50
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

13.7 JEUX D'ESSAI : recopier une phrase en inversant chaque mot.(procédure inverser_mot)

Le caractère espace sera remplacé ici par le caractère **■**.

- | | | |
|----|--|-----------------------------------|
| 1) | texte = 'titi.'
ind = 5
longmot = 4
inverse = "
indinv = 1 | inverse = 'titi'
indinv = 5 |
| 2) | texte = 'le■chat■.'
ind = 8
longmot = 4
inverse = 'el■'
indinv = 4 | inverse = 'el■tahc'
indinv = 8 |
| 3) | texte = "
ind = 0
longmot = 0
inverse = 'tutu'
indinv = 5 | inverse = 'tutu'
indinv = 5 |
| 4) | texte = 'abcdefg.'
ind = 6
longmot = 3
inverse = 'a'
indinv = 2 | inverse = 'adec'
indinv = 5 |

13.8 SOLUTION : *recopier une phrase en inversant chaque mot.(procédure inverser_mot)*

Procédure inverser_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
Entrée Sortie indinv : entier, Entrée Sortie inverse : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position
 // " après ", en inversant les lettres du mot, à l'emplacement indiqué
 // par indinv dans la chaîne inverse

// texte est la chaîne où se trouve le mot
 // ind est la position dans texte juste après le mot
 // longmot est la longueur du mot
 // indinv est en entrée l'endroit où recopier le mot inversé
 // en sortie l'endroit où mettre l'espace ou le point final
 // inverse est en entrée la portion de chaîne qui a déjà été inversée
 // en sortie le mot est recopié à la suite

Début

// indice positionné sur le dernier caractère du mot de la phrase

ind := ind - 1

// parcours du mot à l'envers en recopiant lettre à lettre

Tantque (longmot <> 0) **Faire**
 // arrêt quand le mot est parcouru

inverse [indinv] := texte [ind]
 indinv := indinv + 1
 ind := ind - 1
 longmot := longmot - 1

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 52
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

14. EXERCICE : justifier une phrase.

Soit une phrase de 80 caractères terminée par un point.

Donnez l'algorithme du programme qui justifie cette phrase. La justification d'une phrase consiste à répartir les mots sur la totalité de la phrase en répartissant équitablement les espaces entre les mots. S'il reste des espaces ils seront rajoutés aux premiers intervalles.

14.1 SOLUTION: *justifier une phrase.(algorithme de principe)*

prendre un mot

Tantque la longueur du mot est non nulle **Faire**

compter les mots

compter le nombre total des lettres des mots

prendre un mot

Fintantque

calculer le nombre d'espaces à mettre entre les mots

calculer le nombre d'espaces en trop

Tantque tous les mots ne sont pas copiés **Faire**

prendre un mot

copier le mot

Si ce n'est pas le dernier mot **Alors**

mettre les espaces

Si il reste des espaces en trop **Alors**

mettre un espace

décrémenter le nombre d'espaces en trop

Finsi

Finsi

Fintantque

mettre le point final

Définition des données:

un mot de la phrase est donné par :

la phrase initiale

l'indice juste après le mot ou sur le caractère terminateur

la longueur du mot ou 0 si le mot est le caractère terminateur

La phrase est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm

La phrase résultat est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 53
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

14.2 SOLUTION : *justifier une phrase.(interfaces)*

Procédure justifier (Entrée texte : chaîne , Sortie texte_jus : chaîne)

// Cette procédure justifie une chaîne de caractères dans une autre chaîne.

// texte est la chaîne de caractères à traiter

// texte_jus est le résultat de la justification de texte

Procédure prendre_mot (Entrée texte : chaîne, Entrée Sortie ind : entier, Sortie lg : entier)

// Cette procédure positionne un indice sur le caractère suivant le mot

// repéré, et donne sa longueur, ou positionne l'indice sur le caractère

// terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot

// ind est en entrée la position à partir de laquelle on cherche un mot

// en sortie la position juste après le mot ou carterm

// lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure copier_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
Entrée Sortie indjus : entier, Entrée Sortie texte_jus : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position

// " après ", à l'emplacement indiqué par indjus dans la chaîne texte_jus

// texte est la chaîne où se trouve le mot

// ind est la position dans texte juste après le mot

// longmot est la longueur du mot

// indjus est en entrée l'endroit où recopier le mot

// en sortie l'endroit où mettre l'espace ou le point final

// texte_jus est en entrée la portion de chaîne qui a déjà été justifiée

// en sortie le mot est recopié à la suite

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 54
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

14.3 SOLUTION : *justifier une phrase.(programme)*

<u>Programme</u>	justifier_phrase
------------------	------------------

// Ce programme justifie une phrase dans une autre, c'est à dire réparti les mots sur la
// totalité de la longueur de la phrase en plaçant harmonieusement les espaces.

Constantes taille = 25 // nombre maximum de caractères dans la chaîne
 carterm = '.' // caractère terminant la phrase à traiter

Types chaîne = **tableau** [taille] **de caractères**
// type des chaînes de caractères traitées

Variables phrase : chaîne // phrase est la chaîne où l'on cherche le mot
 phrase_résultat : chaîne // phrase_résultat est la phrase justifiée

Procédure justifier (**Entrée** texte : chaîne , **Sortie** texte_jus : chaîne)

```
// Cette procédure justifie une chaîne de caractères dans une autre
// chaîne.
```

```
// texte est la chaîne de caractères à traiter
// texte_jus est le résultat de la justification de texte
```

Début // saisie de la phrase à traiter

Ecrire ('donnez la phrase terminée par un caractère ',carterm, ' : ')

Lire (phrase)

```
// appel de la procédure qui fait tout le travail
```

justifier (phrase, phrase_résultat)

// résultat du travail

Ecrire ('la phrase est : ', phrase)

Ecrire ('la phrase justifiée est : ', phrase_résultat)

Fin

14.4 JEUX D'ESSAI : justifier une phrase. (procédure justifier)

Le caractère espace est représenté ici par le caractère α .

- | | |
|--------------------------------------|--|
| 1) phrase = ':' | phrase_résultat = ':' |
| 2) phrase = 'aaa.' | phrase_résultat = ':' |
| 3) phrase = 'aaatrop.' | phrase_résultat = 'trop.' |
| 4) phrase = 'aaaleaachataaaa.' | phrase_résultat = 'leaaaaaaaaaaaaaaaaaaaaaachata.' |
| 5) phrase = 'leaachatæstægris.' | phrase_résultat = 'leaaaaaachataaaaaæstaaaagris.' |
| 6) phrase = 'aaaleaachataæstægrisα.' | phrase_résultat = 'leaaaaaachataaaaaæstaaaagris.' |

14.5 SOLUTION : justifier une phrase. (procédure justifier)

Procédure justifier (Entrée texte : chaîne , Sortie texte_jus : chaîne)

// Cette procédure justifie une chaîne de caractères dans une autre chaîne.

// texte est la chaîne de caractères à traiter

// texte_jus est le résultat de la justification de texte

Constantes espace = ' ' // caractère séparateur de mots

Variables

i_t	: entier	// indice de parcours de la chaîne texte
i_j	: entier	// indice de parcours de la chaîne justifiée
taille_mot	: entier	// longueur d'un mot repéré dans le texte
nb_mots	: entier	// nombre de mots dans le texte
nb_lettres	: entier	// nombre de caractères utiles du texte
intervalle	: entier	// nombre d'espaces à mettre entre les mots
reste	: entier	// espaces à répartir harmonieusement
nb_espaces	: entier	// nombre d'espaces à mettre après le mot copié

Procédure prendre_mot (Entrée texte : chaîne, Entrée Sortie ind : entier, Sortie lg : entier)

// Cette procédure positionne un indice sur le caractère suivant le mot
// repéré, et donne sa longueur, ou positionne l'indice sur le caractère
// terminateur, et donne la longueur 0.

// texte est la chaîne de caractère où l'on veut repérer un mot
// ind est en entrée la position à partir de laquelle on cherche un mot
// en sortie la position juste après le mot ou carterm
// lg est la longueur du mot repéré ou 0 si il n'y a plus de mot

Procédure copier_mot (Entrée texte : chaîne , Entrée ind : entier, Entrée longmot : entier,
Entrée Sortie indjus : entier, Entrée Sortie texte_jus : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position
// " après ", à l'emplacement indiqué par indjus dans la chaîne texte_jus

// texte est la chaîne où se trouve le mot
// ind est la position dans texte juste après le mot
// longmot est la longueur du mot
// indjus est en entrée l'endroit où recopier le mot
// en sortie l'endroit où mettre l'espace ou le point final
// texte_jus est en entrée la portion de chaîne qui a déjà été justifiée
// en sortie le mot est recopié à la suite

Début // indice au début de la chaîne texte et initialisation des compteurs

i_t := 1
nb_mots := 0
nb_lettres := 0

// parcours de la phrase pour compter les mots et les caractères utiles

prendre_mot (texte, i_t, taille_mot)

Tantque taille_mot <> 0 **Faire** // arrêt quand on a détecté le terminateur
 nb_mots := nb_mots + 1 // comptage des mots
 nb_lettres := nb_lettres + taille_mot // comptage des caractères utiles
 prendre_mot (texte, i_t, taille_mot) // repérage d'un mot dans le texte

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 57
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

Fintantque

// calcul d'intervalle d'espaces, et le reste à répartir entre les mots

Si nb_mots > 1 **Alors** // il y a plusieurs mots, on peut donc calculer
 intervalle := (taille - nb_lettres - 1) **div** (nb_mots - 1)
 reste := (taille - nb_lettres - 1) **mod** (nb_mots - 1)

Finsi

// initialisation des indices au début des phrases

i_t := 1
 i_j := 1

// parcours du texte en recopiant le texte justifié

Tantque nb_mots <> 0 **Faire** // arrêt quand on a traité tous les mots

prendre_mot (texte, i_t, taille_mot)
 copier_mot (texte, i_t, taille_mot, i_j, texte_jus)
 nb_mots := nb_mots - 1

// traitement des espaces à mettre entre les mots

Si nb_mots <> 0 **Alors** // il reste des mots, il faut mettre des espaces

// calcul du nombre d'espaces à mettre entre les mots

nb_espaces := intervalle

Si reste <> 0 **Alors** // il reste des espaces en trop
 reste := reste - 1
 nb_espaces := nb_espaces + 1

Finsi

// insertion d'espaces entre les mots

Tantque nb_espaces <> 0 **Faire**
 // arrêt quand il n'y a plus d'espaces à copier
 texte_jus [i_j] := espace
 i_j := i_j + 1
 nb_espaces := nb_espaces - 1

Fintantque**Finsi****Fintantque**

// terminateur mis dans la phrase résultat

texte_jus [i_j] := carterm

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 58
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

14.6 JEUX D'ESSAI : justifier une phrase.(procédure prendre_mot)

Le caractère espace sera ici représenté par le caractère **␣**.

- | | |
|---|--|
| 1) <i>texte</i> = '.'
<i>ind</i> = 1 | <i>longueur</i> = 0
<i>ind</i> = 1 |
| 2) <i>texte</i> = '␣␣␣.'
<i>ind</i> = 1 | <i>longueur</i> = 0
<i>ind</i> = 4 |
| 3) <i>texte</i> = 'le␣chat␣est␣gris.'
<i>ind</i> = 1 | <i>longueur</i> = 2
<i>ind</i> = 3 |
| 4) <i>texte</i> = '␣␣␣le␣chat.'
<i>ind</i> = 1 | <i>longueur</i> = 2
<i>ind</i> = 6 |
| 5) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 3 | <i>longueur</i> = 4
<i>ind</i> = 9 |
| 6) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 13 | <i>longueur</i> = 4
<i>ind</i> = 18 |
| 7) <i>texte</i> = 'le␣␣chat␣est␣gris␣␣.'
<i>ind</i> = 13 | <i>longueur</i> = 4
<i>ind</i> = 18 |
| 8) <i>texte</i> = 'le␣␣chat␣est␣gris.'
<i>ind</i> = 18 | <i>longueur</i> = 0
<i>ind</i> = 18 |
| 9) <i>texte</i> = 'le␣␣chat␣est␣gris␣␣.'
<i>ind</i> = 18 | <i>longueur</i> = 0
<i>ind</i> = 20 |

14.7 SOLUTION : *justifier une phrase.(procédure prendre_mot)*

Procédure prendre_mot (**Entrée** texte : chaîne, **Entrée Sortie** ind : **entier**, **Sortie** lg : **entier**)

```
// Cette procédure positionne un indice sur le caractère suivant le mot
// repéré, et donne sa longueur, ou positionne l'indice sur le caractère
// terminateur, et donne la longueur 0.
```

```
// texte est la chaîne de caractère où l'on veut repérer un mot
// ind est      en entrée la position à partir de laquelle on cherche un mot
//              en sortie la position juste après le mot ou cartermin
// lg est la longueur du mot repéré ou 0 si il n'y a plus de mot
```

Début // éventuels espaces en début de mot sautés

Tantque texte [ind] = espace **Faire** // arrêt quand on a autre chose qu'un espace

$$\text{ind} := \text{ind} + 1$$

Fintantque

```
// initialisation de la longueur
```

$$\lg := 0$$

```
// recherche de la fin du mot
```

Tantque (texte [ind] <> espace) **et** (texte [ind] <> carterm) **Faire**
 // arrêt quand on a un espace ou le carterm

```
ind := ind + 1
lg := lg + 1
```

Fintantque

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 60
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

14.8 JEUX D'ESSAI : justifier une phrase. (procédure copier)

Le caractère espace sera remplacé ici par le caractère α .

```
1) texte = 'titi.'  
ind = 5  
longmot = 4  
texte_jus = '  
indjus = 1
```

```
texte_jus = 'titi'
indjus = 5
```

```
2) texte = 'leachat.'
   ind = 8
   longmot = 4
   texte_jus = 'leoooooooooooooooooooooooooooo'
   indjus = 21
```

```
texte_jus = 'leoooooooooooooooooooochat'
indjus = 25
```

```
3) texte = ''
   ind = 0
   longmot = 0
   texte_jus = 'tutu'
   indjus = 5
```

```
texte_jus = 'tutu'  
indjus = 5
```

```
4) texte = 'abcdefg.'
```

```
texte_jus = 'aced'
indjus = 5
```

Les exemples 3 et 4 ne sont pas des utilisations de la procédure copier dans le cadre de la justification, mais ils sont des exemples de test de la procédure.

14.9 SOLUTION : *justifier une phrase.(procédure copier)*

Procédure copier_mot (**Entrée** texte : chaîne , **Entrée** ind : entier, **Entrée** longmot : entier,
Entrée Sortie indjus : entier, **Entrée Sortie** texte_jus : chaîne)

// Cette procédure recopie un mot repéré dans la chaîne par sa position
 // " après ", à l'emplacement indiqué par indjus dans la chaîne texte_jus

 // texte est la chaîne où se trouve le mot
 // ind est la position dans texte juste après le mot
 // longmot est la longueur du mot
 // indjus est en entrée l'endroit où recopier le mot
 // en sortie l'endroit où mettre l'espace ou le point final
 // texte_jus est en entrée la portion de chaîne qui a déjà été justifiée
 // en sortie le mot est recopié à la suite

Début

// indice positionné sur le premier caractère du mot à recopier

ind := ind - longmot

// parcours du mot en recopiant lettre à lettre

Tantque (longmot <> 0) **Faire** // arrêt quand le mot est parcouru

texte_jus [indjus] := texte [ind]
 indjus := indjus + 1
 ind := ind + 1
 longmot := longmot - 1

Fintantque**Fin**

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 62
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

15. EXERCICE : calculer la somme de nombres en base quelconque.

Soit une chaîne de caractères représentant des nombres dans une base quelconque comprise entre 2 et 16.
Soit b la base dans laquelle ces nombres sont écrits.

Le premier caractère zéro rencontré en début de nombre indique la fin de la chaîne de caractères à traiter.

Donnez l'algorithme de la procédure qui, à partir de ces informations, donne l'entier représentant la somme des nombres contenus dans cette chaîne.

Exemple :

1	7		2	0	3			9		0	
---	---	--	---	---	---	--	--	---	--	---	--

base $b = 13$

L'algorithme donnera un entier de valeur $1 \cdot 13 + 7 + 2 \cdot 13 \cdot 13 + 0 \cdot 13 + 3 + 9 = 20 + 341 + 9 = 370$

Note : On sait que les informations contenues dans la chaîne de caractères sont correctes.
Les nombres sont non signés et on ne traite pas les débordements.

15.1 SOLUTION : *calculer la somme de nombres en base quelconque.(algorithme de principe)*

Somme := 0

Répéter

chercher le début d'un nombre

Si le premier caractère est différent du caractère terminateur **Alors**

parcourir en décodant le nombre

somme := somme + nombre

Finsi

jusqu'à être positionné sur le caractère terminateur

Algorithme de principe de parcourir en décodant le nombre:

nombre := 0

Répéter

convertir le caractère dans la base considérée

nombre := nombre * base + caractère converti

passer au caractère suivant

Jusqu'à arriver sur un caractère espace

Définition des données:

un début d'un nombre est donné par :

la phrase initiale

l'indice du premier caractère du nombre ou du caractère terminateur

La phrase est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm en début de nombre

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 63
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

15.2 SOLUTION : calculer la somme de nombres en base quelconque.(interfaces)

Procédure calculer_somme (Entrée texte : chaîne, Entrée base : entier, Sortie résultat : entier)

// Cette procédure calcule la somme des nombres contenus dans une
 // chaîne de caractères. Les nombres sont représentés dans une base
 // quelconque de 2 à 16. Les nombres sont séparés par au moins un espace.
 // La chaîne est terminée par un zéro en début de nombre

// texte est la chaîne de caractères à traiter
 // base est la base dans laquelle sont écrits les nombres
 // résultat est la somme des nombres contenus dans la chaîne

Procédure parcourir_décoder (Entrée texte : chaîne , Entrée Sortie ind : entier,
Entrée base : entier, Sortie valeur : entier)

// Cette procédure calcule la valeur d'un nombre à partir de la position
 // courante, dans la base donnée, et se positionne après le nombre

// texte est la chaîne où se trouve le mot
 // ind est en entrée la position du premier chiffre du nombre
 // en sortie la position sur le premier espace après le nombre
 // base est la base d'écriture du nombre
 // valeur est la valeur entière du nombre

Fonction convertir (Entrée car : caractère) : entier

// Cette fonction donne la valeur d'un chiffre

// car est le caractère dont on veut connaître la valeur
 // la fonction retourne la valeur du chiffre

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 64
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

15.3 SOLUTION : calculer la somme de nombres en base quelconque.(programme)

Programme sommer_nombres

// Ce programme calcule la somme des nombres contenus dans une chaîne.
 // Ces nombres sont représentés dans une base de 2 à 16.

Constantes taille = 80 // nombre maximum de caractères dans la chaîne
 carterm = '0' // caractère terminant la phrase
 // à traiter si il est en début de nombre
 espace = ' ' // caractère séparateur de nombres

Types chaîne = **tableau** [taille] **de caractères**
 // type des chaînes de caractères traitées

Variables phrase : chaîne // phrase est la chaîne où l'on cherche le mot
 base_choisie : **entier** // base_choisie est la base de représentation des nombres
 résultat : **entier** // résultat est la somme des nombres de la phrase

Procédure calculer_somme (**Entrée** texte : chaîne, **Entrée** base : **entier**, **Sortie** résultat : **entier**)

// Cette procédure calcule la somme des nombres contenus dans une
 // chaîne de caractères. Les nombres sont représentés dans une base
 // quelconque de 2 à 16. Les nombres sont séparés par au moins un espace.
 // La chaîne est terminée par un zéro en début de nombre

// texte est la chaîne de caractères à traiter
 // base est la base dans laquelle sont écrits les nombres
 // résultat est la somme des nombres contenus dans la chaîne

Début

// saisie de la base et de la phrase à traiter

Ecrire ('donnez la suite de nombres terminée par un zéro en début de nombre : ')

Lire (phrase)

Ecrire ('donnez la base de représentation des nombres : ')

Lire (base_choisie)

// appel de la procédure qui fait tout le travail

calculer_somme (phrase, base_choisie, résultat)

// résultat du travail

Ecrire ('la somme des nombres (exprimée en base dix) est: ', résultat)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 65
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

15.4 JEUX D'ESSAI : calculer la somme de nombres en base quelconque.(procédure calculer_somme)

Le caractère espace est représenté ici par le caractère π .

- | | |
|---|----------------|
| 1) phrase = '0'
base_choisie = 7 | résultat = 0 |
| 2) phrase = ' $\pi\pi\pi$ 0'
base_choisie = 9 | résultat = 0 |
| 3) phrase = '123 π 0'
base_choisie = 4 | résultat = 27 |
| 4) phrase = ' $\pi\pi\pi$ 123 $\pi\pi\pi$ 0'
base_choisie = 4 | résultat = 27 |
| 5) phrase = '123 π 0'
base_choisie = 5 | résultat = 38 |
| 6) phrase = '2b π 0'
base_choisie = 16 | résultat = 43 |
| 7) phrase = '102 π 0'
base_choisie = 6 | résultat = 38 |
| 8) phrase = '53 π 0'
base_choisie = 7 | résultat = 38 |
| 9) phrase = ' π 24 $\pi\pi$ 13 $\pi\pi$ 9 $\pi\pi$ 17 π 0'
base_choisie = 10 | résultat = 63 |
| 10) phrase = '45 π 23 π 1 π 10 π 0'
base_choisie = 6 | résultat = 51 |
| 11) phrase = 'ab π 3c $\pi\pi\pi$ 123 π 0'
base_choisie = 14 | résultat = 432 |

15.5 SOLUTION : calculer la somme de nombres en base quelconque.(procédure calculer_somme)

Procédure calculer_somme (Entrée texte : chaîne, Entrée base : entier, Sortie résultat : entier)

// Cette procédure calcule la somme des nombres contenus dans une
// chaîne de caractères. Les nombres sont représentés dans une base
// quelconque de 2 à 16. Les nombres sont séparés par au moins un espace.
// La chaîne est terminée par un zéro en début de nombre

// texte est la chaîne de caractères à traiter
// base est la base dans laquelle sont écrits les nombres
// résultat est la somme des nombres contenus dans la chaîne

Variables i_t : entier // indice de parcours de la chaîne texte
nombre : entier // nombre est la valeur d'un nombre du texte

Procédure parcourir_décoder (Entrée texte : chaîne , Entrée Sortie ind : entier,
Entrée base : entier, Sortie valeur : entier)

// Cette procédure calcule la valeur d'un nombre à partir de la position
// courante, dans la base donnée, et se positionne après le nombre

// texte est la chaîne où se trouve le mot
// ind est en entrée la position du premier chiffre du nombre
// en sortie la position sur le premier espace après le nombre
// base est la base d'écriture du nombre
// valeur est la valeur entière du nombre

Début

// indice mis au début de la chaîne texte et initialisation de la somme

i_t := 1
résultat := 0

// parcours de la phrase de nombre en nombre

Répéter

// recherche du début d'un nombre

Tantque texte [i_t] = espace Faire
// arrêt en début de nombre
i_t := i_t + 1

Fintantque

// calcul du nombre si ce n'est pas la fin de chaîne

Si texte [i_t] <> carterm Alors
parcourir_décoder (texte, i_t, base, nombre)
résultat := résultat + nombre

Finsi

Jusqu'à texte [i_t] = carterm // arrêt quand on rencontre le terminateur

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 67
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

15.6 JEUX D'ESSAI : calculer la somme de nombres en base quelconque.(procédure parcourir_décoder)

Le caractère espace sera ici représenté par le caractère α .

- | | |
|--|---|
| 1) $\text{texte} = '1\alpha 0'$
$\text{ind} = 1$
$\text{base} = 10$ | $\text{valeur} = 1$
$\text{ind} = 2$ |
| 2) $\text{texte} = '13\alpha 0'$
$\text{ind} = 1$
$\text{base} = 10$ | $\text{valeur} = 13$
$\text{ind} = 3$ |
| 3) $\text{texte} = '\alpha\alpha\alpha 102\alpha 0'$
$\text{ind} = 4$
$\text{base} = 10$ | $\text{valeur} = 102$
$\text{ind} = 7$ |
| 4) $\text{texte} = '14\alpha\alpha\alpha 56\alpha\alpha 0'$
$\text{ind} = 6$
$\text{base} = 10$ | $\text{valeur} = 56$
$\text{ind} = 8$ |
| 5) $\text{texte} = '\alpha\alpha 4ac\alpha a 7d\alpha\alpha 0'$
$\text{ind} = 7$
$\text{base} = 16$ | $\text{valeur} = 2685$
$\text{ind} = 10$ |
| 6) $\text{texte} = '1101\alpha 100\alpha\alpha 1\alpha 10001\alpha\alpha 0'$
$\text{ind} = 13$
$\text{base} = 2$ | $\text{valeur} = 17$
$\text{ind} = 18$ |
| 7) $\text{texte} = '\alpha 666\alpha 4532\alpha 0'$
$\text{ind} = 6$
$\text{base} = 7$ | $\text{valeur} = 1640$
$\text{ind} = 10$ |

15.7 SOLUTION : calculer la somme de nombres en base quelconque.(procédure parcourir_décoder)

Procédure parcourir_décoder (Entrée texte : chaîne , Entrée Sortie ind : entier,
Entrée base : entier, Sortie valeur : entier)

// Cette procédure calcule la valeur d'un nombre à partir de la position
 // courante, dans la base donnée, et se positionne après le nombre

// texte est la chaîne où se trouve le mot
 // ind est en entrée la position du premier chiffre du nombre
 // en sortie la position sur le premier espace après le nombre
 // base est la base d'écriture du nombre
 // valeur est la valeur entière du nombre

Fonction convertir (Entrée car : caractère) : entier

// Cette fonction donne la valeur d'un chiffre

// car est le caractère dont on veut connaître la valeur
 // la fonction retourne la valeur du chiffre

Début

// initialisation de la valeur du nombre

valeur := 0

// conversion jusqu'à trouver un espace

Répéter

valeur := valeur * base + convertir (texte [ind])
 ind := ind + 1

Jusqu'à texte [ind] = espace // arrêt en fin de nombre

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 69
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

15.8 JEUX D'ESSAI : calculer la somme de nombres en base quelconque.(fonction convertir)

Il suffit de contrôler tous les cas de figure (ici en base 16):

0
1
2
3
4
5
6
7
8
9
a
A
b
B
c
C
d
D
e
E
f
F

Il n'y a pas de cas d'erreur, la chaîne de caractères étant supposée correcte en entrée.

afpa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 70
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

15.9 SOLUTION : calculer la somme de nombres en base quelconque.(fonction convertir)

Fonction convertir (Entrée car : caractère) : entier

// Cette fonction donne la valeur d'un chiffre

// car est le chiffre dont on veut connaître la valeur

// la fonction retourne la valeur du chiffre

Variables val_chiffre : entier // val_chiffre est la valeur entière du chiffre

Début // retour de la valeur adéquate suivant la valeur du caractère

choix sur car Faire

```
'0' : val_chiffre := 0
'1' : val_chiffre := 1
'2' : val_chiffre := 2
'3' : val_chiffre := 3
'4' : val_chiffre := 4
'5' : val_chiffre := 5
'6' : val_chiffre := 6
'7' : val_chiffre := 7
'8' : val_chiffre := 8
'9' : val_chiffre := 9
'a','A' : val_chiffre := 10
'b','B' : val_chiffre := 11
'c','C' : val_chiffre := 12
'd','D' : val_chiffre := 13
'e','E' : val_chiffre := 14
Autrecas : val_chiffre := 15
```

Finchoix

// retour du résultat de la fonction

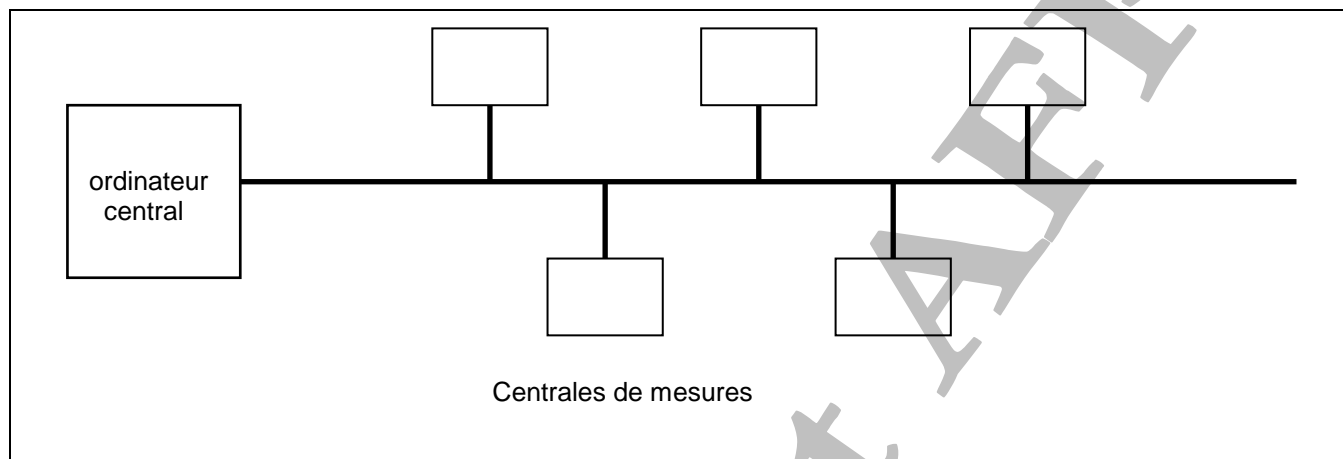
retourner (val_chiffre)

Fin

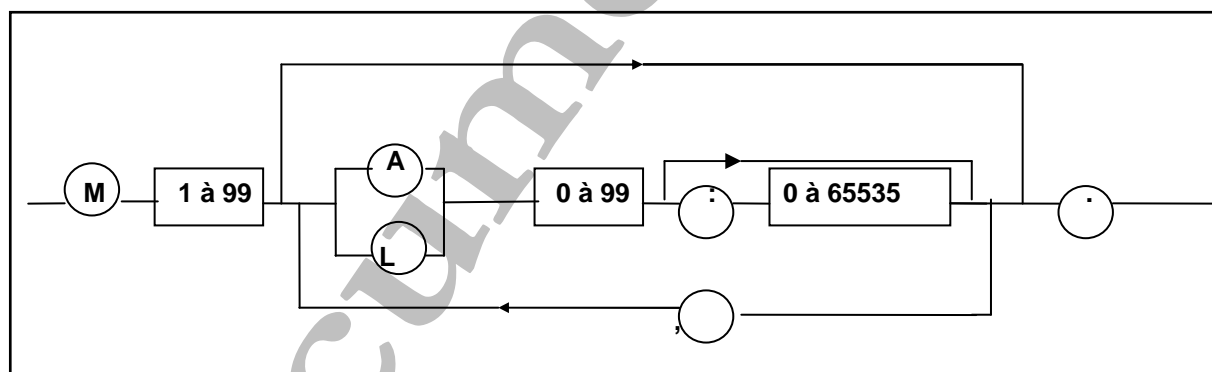
afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 71
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

16. EXERCICE : vérifier la syntaxe d'une commande.

Un ordinateur central est relié à des centrales de mesures (machines permettant de lire et d'écrire des valeurs logiques et analogiques). Ces centrales sont reliées en grappe à l'ordinateur central. Les centrales de mesure sont au maximum 99. Elles ont au maximum 100 voies analogiques ou logiques. Les voies logiques permettent 16 entrées/sorties tout ou rien. Les voies analogiques permettent de donner des consignes ou de lire des valeurs de 0 à 10 Volts et la conversion analogique/numérique se fera sur 16 bits.



Voici la syntaxe du langage permettant de communiquer avec les centrales de mesure depuis le serveur.



Donnez l'algorithme de la procédure qui permet de savoir si la syntaxe de la commande envoyée à une centrale est correcte ou non.

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 72
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

16.1 SOLUTION : vérifier la syntaxe d'une commande. (algorithme de principe)

erreur := **faux**

Si la première lettre est 'M' **Alors**

passage au caractère suivant

Si on a un nombre entre 1 et 99 **Alors**

Si le caractère est 'A' ou 'L' **Alors**

analyse d'une voie

Tantque il n'y a pas d'erreur et que la lettre est ',' **Faire**

passage au caractère suivant

analyse d'une voie

Fintantque

Finsi

Si le caractère n'est pas le '.' **Alors**

erreur := **vrai**

Finsi

Sinon

erreur := **vrai**

Finsi

Sinon

erreur := **vrai**

Finsi

Algorithme de principe de l'analyse d'une voie :

Si le caractère est 'A' ou 'L' **Alors**

passage au caractère suivant

Si on a un nombre entre 0 et 99 **Alors**

Si le caractère est un ':' **Alors**

passage au caractère suivant

Si on n'a pas un nombre entre 0 et 65535 **Alors**

erreur := **vrai**

Finsi

Finsi

Sinon

erreur := **vrai**

Sinon

erreur := **vrai**

Finsi

Note:

- chaque procédure fait uniquement le travail qui lui est dévolu.
- chaque procédure doit à la sortie se positionner sur le caractère suivant les caractères traités.

Définition des données:

La phrase est donnée par :

une chaîne de caractères de type chaîne

un caractère terminateur carterm

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 73
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

16.2 SOLUTION : vérifier la syntaxe d'une commande.(interfaces)

Procédure vérifier_syntaxe (Entrée texte : chaîne, Sortie erreur : booléen)

// Cette procédure vérifie la syntaxe d'une commande destinée à une centrale de mesure

// texte est la chaîne de caractères à traiter

// erreur sera vrai si la syntaxe de texte n'est pas conforme

Procédure analyser_voie (Entrée texte : chaîne , Entrée Sortie ind : entier,
Sortie erreurvoie : booléen)

// Cette procédure vérifie la syntaxe d'une voie de la commande d'une centrale de mesure

// texte est la chaîne où se trouve la commande

// ind est en entrée la position sur une voie de la commande

// en sortie la position sur le premier caractère après la voie

// erreurvoie est vrai si la syntaxe de la voie n'est pas correcte

Fonction vérifier_nombre (Entrée texte : chaîne, Entrée Sortie ind : entier,
Entrée min : entier, Entrée max : entier) : booléen

// Cette fonction dit si un nombre est bien à l'emplacement indiqué

// et si ce nombre est compris, au sens large, entre les bornes

// texte est la chaîne où se trouve la commande

// ind est en entrée la position sur le premier chiffre du nombre

// en sortie la position sur le caractère après le nombre

// min est la borne minimum pour le nombre

// max est la borne maximum pour le nombre

// texte est la chaîne où se trouve la commande

// la fonction retourne vrai si c'est bien un nombre, qui est dans les bornes

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 74
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

16.3 SOLUTION : vérifier la syntaxe d'une commande.(programme)

Programme vérifier_syntaxe_commande

// Ce programme vérifie qu'une chaîne de caractères vérifie la syntaxe des commandes
 // de centrales de mesure.

Constantes taille = 80 // nombre maximum de caractères dans la chaîne

Types chaîne = **tableau** [taille] **de caractères**
 // type des chaînes de caractères traitées

Variables phrase : chaîne // phrase est la chaîne qui contient la commande
 résultat : **booléen** // résultat est faux quand phrase vérifie la syntaxe

Procédure vérifier_syntaxe (**Entrée** texte : chaîne, **Sortie** erreur : **booléen**)

// Cette procédure vérifie la syntaxe d'une commande destinée à une centrale de mesure

// texte est la chaîne de caractères à traiter
 // erreur sera vrai si la syntaxe de texte n'est pas conforme

Début

// saisie de la phrase à traiter

Ecrire ('donnez la commande à envoyer à une centrale de mesure : ')

Lire (phrase)

// appel de la procédure qui fait tout le travail

vérifier_syntaxe (phrase, résultat)

// résultat du travail

Si résultat **Alors**
 Ecrire ('la syntaxe de la commande n'est pas correcte.')

Sinon
 Ecrire ('la syntaxe de la commande est correcte.')

Finsi

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 75
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

16.4 JEUX D'ESSAI : vérifier la syntaxe d'une commande.(procédure vérifier_syntaxe)

Le caractère espace est représenté ici par le caractère α .

1) phrase = '.'	erreur = vrai
2) phrase = 'b2.'	erreur = vrai
3) phrase = 'M2.'	erreur = faux
4) phrase = 'M2,.'	erreur = vrai
5) phrase = 'M0.'	erreur = vrai
6) phrase = 'M100.'	erreur = vrai
7) phrase = 'M4A5.'	erreur = faux
8) phrase = 'M4A100.'	erreur = vrai
9) phrase = 'M4L25.'	erreur = faux
10) phrase = 'M4L.'	erreur = vrai
11) phrase = 'M4J.'	erreur = vrai
12) phrase = 'M4L2:.'	erreur = vrai
13) phrase = 'M4L2:,A3.'	erreur = vrai
14) phrase = 'M4L2:34.'	erreur = faux
15) phrase = 'M4L2:67899.'	erreur = vrai
16) phrase = 'M4L2:567,.'	erreur = vrai
17) phrase = 'M4L2:676,A4.'	erreur = faux
18) phrase = 'M4L2:657,B6.'	erreur = vrai
19) phrase = 'M4L2:66,A7:42,L6:89.'	erreur = faux
20) phrase = 'M4L2:45,A5,L2.'	erreur = faux
21) phrase = 'M4L2:45,A5,L2L.'	erreur = vrai

16.5 SOLUTION : vérifier la syntaxe d'une commande.(procédure vérifier_syntaxe)

Procédure vérifier_syntaxe (Entrée texte : chaîne, Sortie erreur : booléen)

// Cette procédure vérifie la syntaxe d'une commande destinée à une centrale de mesure

// texte est la chaîne de caractères à traiter

// erreur sera vrai si la syntaxe de texte n'est pas conforme

Constantes

machine	= 'M'	// entête de commande
analogique	= 'A'	// voie analogique
logique	= 'L'	// voie logique
écriture_valeur	= ':'	// écriture d'une valeur sur la voie concernée
voie_suivante	= ','	// séparateur de commande de voie
fin_commande	= '.'	// caractère terminant la commande
centrale_min	= 1	// numéro minimum d'une centrale
centrale_max	= 99	// numéro maximum d'une centrale
voie_min	= 0	// numéro minimum d'une voie
voie_max	= 99	// numéro maximum d'une voie
valeur_min	= 0	// valeur minimum à écrire sur une voie
valeur_max	= 65535	// valeur maximum à écrire sur une voie

Variables i_t : entier // indice de parcours de la chaîne texte

Procédure analyser_voie (Entrée texte : chaîne , Entrée Sortie ind : entier,
Sortie erreurvoie : booléen)

// Cette procédure vérifie la syntaxe d'une voie de la commande d'une centrale de mesure

// texte est la chaîne où se trouve la commande

// ind est en entrée la position sur une voie de la commande

// en sortie la position sur le premier caractère après la voie

// erreurvoie est vrai si la syntaxe de la voie n'est pas correcte

Fonction vérifier_nombre (Entrée texte : chaîne, Entrée Sortie ind : entier,
Entrée min : entier, Entrée max : entier) : booléen

// Cette fonction dit si c'est bien un nombre à l'emplacement indiqué

// et si ce nombre est compris, au sens large, entre les bornes

// texte est la chaîne où se trouve la commande

// ind est en entrée la position sur le premier chiffre du nombre

// en sortie la position sur le caractère après le nombre

// min est la borne minimum pour le nombre

// max est la borne maximum pour le nombre

// texte est la chaîne où se trouve la commande

// la fonction retourne vrai si c'est bien un nombre, qui est dans les bornes

Début // indice au début de la chaîne texte et initialisation erreur

i_t := 1

erreur := faux

// parcours de la commande en vérifiant la syntaxe

Si texte [i_t] = machine **Alors** // ça commence bien

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 77
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

```

i_t := i_t + 1
Si Vérifier_nombre ( texte, i_t, centrale_min, centrale_max ) Alors
    // le numéro de centrale est correct

    Si ( texte [ i_t ] = analogique ) ou ( texte [ i_t ] = logique ) Alors
        // commandes de voies repérées

        analyser_voie ( texte, i_t, erreur )
        Tantque ( non erreur ) et ( texte [ i_t ] = voie_suivante ) Faire
            // arrêt quand il y a une erreur dans la voie
            // ou quand il n'y a plus de voie à traiter

            Si i_t = taille Alors
                erreur := vrai

            Sinon
                i_t := i_t + 1
                analyser_voie ( texte, i_t, erreur )

            Finsi

        Fintantque

    Finsi
    Si texte [ i_t ] <> fin_commande Alors
        // attente du caractère de fin de commande
        erreur := vrai

    Finsi

Sinon // le numéro de centrale est incorrect
    erreur := vrai

Finsi

Sinon // ça commence mal
    erreur := vrai

Finsi

Fin

```

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 78
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

16.6 JEUX D'ESSAI : vérifier la syntaxe d'une commande.(procédure analyser_voie)

- | | |
|--|--|
| 1) <i>texte</i> = 'M2A1.' | <i>erreurvoie</i> = faux
<i>ind</i> = 5 |
| 2) <i>texte</i> = 'M2A1B.' | <i>erreurvoie</i> = faux
<i>ind</i> = 5 |
| 3) <i>texte</i> = 'M2A1,B.' | <i>erreurvoie</i> = faux
<i>ind</i> = 5 |
| 4) <i>texte</i> = 'M2A100.' | <i>erreurvoie</i> = vrai
<i>ind</i> = 7 |
| 5) <i>texte</i> = 'M2A1:,A2.' | <i>erreurvoie</i> = vrai
<i>ind</i> = 6 |
| 6) <i>texte</i> = 'M2A1:12345.' | <i>erreurvoie</i> = faux
<i>ind</i> = 11 |
| 7) <i>texte</i> = 'M2A1:999999,A2,A3,L4:89.' | <i>erreurvoie</i> = vrai
<i>ind</i> = 12 |

16.7 SOLUTION : vérifier la syntaxe d'une commande. (procédure analyser_voie)

Procédure analyser_voie (Entrée texte : chaîne , Entrée Sortie ind : entier,
Sortie erreurvoie : booléen)

// Cette procédure vérifie la syntaxe d'une voie de la commande d'une centrale de mesure

// texte est la chaîne où se trouve la commande
 // ind est en entrée la position sur une voie de la commande
 // en sortie la position sur le premier caractère après la voie
 // erreurvoie est vrai si la syntaxe de la voie n'est pas correcte

Fonction vérifier_nombre (Entrée texte : chaîne, Entrée Sortie ind : entier,
Entrée min : entier, Entrée max : entier) : booléen

// Cette fonction dit si c'est bien un nombre à l'emplacement indiqué
 // et si ce nombre est compris, au sens large, entre les bornes

// texte est la chaîne où se trouve la commande
 // ind est en entrée la position sur le premier chiffre du nombre
 // en sortie la position sur le caractère après le nombre
 // min est la borne minimum pour le nombre
 // max est la borne maximum pour le nombre
 // texte est la chaîne où se trouve la commande

// la fonction retourne vrai si c'est bien un nombre, qui est dans les bornes

Début // il n'y a pas à priori d'erreur sur la commande de voie

erreurvoie := faux

// commande de voie examinée

Si (texte [ind] = analogique) ou (texte [ind] = logique) **Alors**
 // la commande de voie commence bien

Si ind = taille **Alors**
 erreurvoie := vrai

Sinon
 ind := ind + 1

Si vérifier_nombre (texte, ind, voie_min, voie_max) **Alors**
 // le numéro de voie est bon

Si texte [ind] = écriture_valeur **Alors**
 // la commande de voie est d'écrire une valeur

Si ind = taille **Alors**
 erreurvoie := vrai

Sinon
 ind := ind + 1

Si non vérifier_nombre (texte, ind, valeur_min, valeur_max) **Alors**
 // la valeur à écrire est incorrecte
 erreurvoie := vrai

Finsi

Finsi

Finsi

afpa®	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 80
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

Sinon // le numéro de voie est incorrect
erreurvoie := **vrai**
Finsi

Finsi
Sinon // le type de la voie est incorrect
erreurvoie := **vrai**
Finsi

Fin

Document AFPA

afpa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 81
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

16.8 JEUX D'ESSAI : vérifier la syntaxe d'une commande.(fonction vérifier_nombre)

1) *texte* = '.'
ind = 1
min = 33
max = 40

vérifier_nombre = **faux**
ind = 1

2) *texte* = 'abcd.'
ind = 1
min = 33
max = 40

vérifier_nombre = **faux**
ind = 1

3) *texte* = 'abc123fgh.'
ind = 4
min = 5
max = 500

vérifier_nombre = **vrai**
ind = 7

4) *texte* = 'df1234567.'
ind = 7
min = 6
max = 500

vérifier_nombre = **faux**
ind = 10

5) *texte* = '12.'
ind = 1
min = 33
max = 40

vérifier_nombre = **faux**
ind = 3

6) *texte* = 'M4A45.'
ind = 4
min = 0
max = 99

vérifier_nombre = **vrai**
ind = 6

16.9 SOLUTION : vérifier la syntaxe d'une commande.(fonction vérifier_nombre)

Fonction vérifier_nombre (Entrée texte : chaîne, Entrée Sortie ind : entier,
Entrée min : entier, Entrée max : entier) : booléen

// Cette fonction dit si c'est bien un nombre à l'emplacement indiqué
 // et si ce nombre est compris, au sens large, entre les bornes

// texte est la chaîne où se trouve la commande
 // ind est en entrée la position sur le premier chiffre du nombre
 // en sortie la position sur le caractère après le nombre
 // min est la borne minimum pour le nombre
 // max est la borne maximum pour le nombre
 // texte est la chaîne où se trouve la commande

// la fonction retourne vrai si c'est bien un nombre, qui est dans les bornes

Constantes base = 10 // base de représentation des nombres

Variables nombre : entier // nombre est la valeur lue dans la chaîne
 ok : booléen // ok sera vrai quand un nombre cohérent sera lu

Fonction convertir (Entrée car : caractère) : entier

// Cette fonction donne la valeur d'un chiffre

// car est le chiffre dont on veut connaître la valeur
 // la fonction retourne la valeur du chiffre

Début // est-ce bien un nombre ?

Si (texte [ind] < '0') **ou** (texte [ind] > '9') **Alors**

// ce n'est pas un nombre

ok := Faux

Sinon

// initialisation de la valeur du nombre

nombre := 0

ok := vrai

// conversion jusqu'à trouver un caractère qui ne soit pas un chiffre

Répéter

nombre := nombre * base + convertir (texte [ind])

Si ind = taille **Alors**

ok := faux

Sinon

ind := ind + 1

Finsi

Jusqu'à (texte [ind] < '0') **ou** (texte [ind] > '9') **ou non** ok

// arrêt en fin de nombre ou de chaîne

// vérification que le nombre est bien dans l'intervalle requis

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 83
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

Si ok **Alors** // si il n'y a pas d'erreur on vérifie les bornes
ok := (nombre \geq min) **et** (nombre \leq max)
Finsi

Finsi

Retourner (ok)

Fin

afpa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 84
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

16.10 JEUX D'ESSAI : vérifier la syntaxe d'une commande.(fonction convertir)

Il suffit de contrôler tous les cas de figure :

0
1
2
3
4
5
6
7
8
9

Il n'y a pas de cas d'erreur, le caractère étant supposé correct en entrée.

16.11 SOLUTION : vérifier la syntaxe d'une commande.(fonction convertir)

Fonction convertir (Entrée car : caractère) : entier

// Cette fonction donne la valeur d'un chiffre

// car est le chiffre dont on veut connaître la valeur

// la fonction retourne la valeur du chiffre

Variables val_chiffre : entier // val_chiffre est la valeur entière du chiffre

Début // retour de la valeur adéquate suivant la valeur du caractère

<u>choix</u>	<u>sur</u>	car	<u>Faire</u>
'0'	:		val_chiffre := 0
'1'	:		val_chiffre := 1
'2'	:		val_chiffre := 2
'3'	:		val_chiffre := 3
'4'	:		val_chiffre := 4
'5'	:		val_chiffre := 5
'6'	:		val_chiffre := 6
'7'	:		val_chiffre := 7
'8'	:		val_chiffre := 8
'9'	:		val_chiffre := 9
<u>Autrecas</u>			: val_chiffre := 0

Finchoix

// retour du résultat de la fonction

retourner (val_chiffre)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 85
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

17. EXERCICE : structurer des données associées à des pièces.

Une usine fabrique des pièces : sphériques, cubiques et cylindriques. Pour chaque pièce produite on veut connaître ses dimensions, sa couleur (jaune, vert, bleu, rouge, orange, mauve), son numéro de série et sa date de fabrication.

Donnez une structure de donnée permettant de stocker toutes les informations sur les pièces produites par l'usine.

17.1 SOLUTION *structurer des données associées à des pièces.*

Constantes taille = 10000 // nombre maximum de pièces produites

Type mois_de_l_annee = (janvier, février, mars, avril, mai, juin, juillet, août, septembre, octobre, novembre, décembre)
// type énuméré des mois de l'année

jour_de_semaine = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche)
// type énuméré des jours de la semaine

colorie = (jaune, vert, bleu, rouge, orange, mauve)
// type énuméré des différentes couleurs des pièces

genre = (cubique, cylindrique, sphérique)
// type énuméré des genres des pièces produites

date = **Enregistrement** // dates où les différents constituants sont dissociés
jour : jour_de_semaine // nom du jour de la semaine
mois : mois_de_l_annee // nom du mois de l'année
année : **entier** // numéro de l'année

Finenregistrement

pièce = **Enregistrement** // informations sur les pièces produites
couleur : colorie // couleur de la pièce produite
numéro : **entier** // numéro de série
datefab : date // date de fabrication de la pièce

Choix forme : genre, **Sur** // forme de la pièce produite
cylindrique : mesure: **Enregistrement** // dimensions des cylindres produits
 hauteur : **réel** // hauteur du cylindre
 rayon : **réel** // rayon du cylindre

Finenregistrement
cubique : coté : **réel** // dimensions des cubes produits
sphérique : rayon : **réel** // dimensions des sphères produites

Finchoix
Finenregistrement

Variables table_pièces : **tableau** [taille] **de** pièce
// tableau pour conserver les caractéristiques des pièces

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 86
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

18. EXERCICE : manipuler une pile d'entiers gérée avec un tableau et un indice.

Une pile d'entiers est une structure de données qui permet de stocker des entiers et de les restituer dans un ordre bien précis.

Ces données seront restituées une à une sur demande selon la règle : dernière entrée, première sortie (LIFO: Last In First Out). Ceci correspond à l'usage habituel d'une pile d'assiettes ou de torchons.

Cette pile est réalisée à l'aide d'un tableau d'entier et d'un indice de parcours.

- 1) Donnez le principe de fonctionnement avec les structures de données de cette pile (ajout et retrait d'un élément) en étudiant particulièrement ce qui se passe quand le tableau est vide et quand il est plein.
- 2) Donnez les interfaces des procédures « init_pile », « empiler », « dépiler ».
- 3) Donnez les algorithmes des trois procédures.

18.1 SOLUTION : *manipuler une pile d'entiers gérée avec un tableau et un indice.(structure des données)*

Constantes taille = 50 // taille du tableau contenant les entiers à empiler

Type pile = **Enregistrement** // type des piles d'entiers
 table : **tableau** [taille] **de entier** // table permet de ranger les informations
 indice : **entier** // indice indique le nombre d'élément dans table

Finenregistrement

18.2 SOLUTION : *manipuler une pile d'entiers gérée avec un tableau et un indice.(principe de fonctionnement)*

La pile est vide quand le nombre d'éléments de la pile vaut 0.

La pile est pleine quand le nombre d'éléments de la pile est « taille ».

« empiler » et « dépiler » sont des fonctions qui retournent un booléen qui indique si la fonction s'est bien déroulée.

Si une pile vide est dépilée, la fonction retourne un compte rendu à faux, et l'entier dépilé est quelconque.

Si une pile pleine est empilée, la fonction retourne un compte rendu à faux, et l'entier à empiler n'est pas empilé.

Dans les autres cas ces deux fonctions retournent des comptes rendus à vrai.

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 87
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

18.3 SOLUTION : *manipuler une pile d'entiers gérée avec un tableau et un indice.(Interfaces)***Procédure** init_pile (**Sortie** lifo : pile)

// Cette procédure permet d'initialiser une pile, c'est à dire qu'elle n'ait aucun élément

// lifo est la pile initialisée

Fonction empiler (**Entrée Sortie** lifo : pile, **Entrée** val : **entier**) : **booléen**

// Cette fonction permet d'insérer un élément dans la pile.

// Si la pile est pleine, la fonction ne fait rien, mais retourne faux.

// lifo est en entrée la pile dans laquelle on veut empiler

// en sortie la pile dans laquelle val est empilé si possible

// val est la valeur à empiler

// la fonction retourne vrai si val a été empilé, faux sinon

Fonction dépiler (**Entrée Sortie** lifo : pile, **Sortie** val : **entier**) : **booléen**

// Cette fonction permet de récupérer l'élément au sommet de la pile.

// Si la pile est vide, la fonction ne fait rien, mais retourne faux

// lifo est en entrée la pile dans laquelle on veut prendre une valeur

// en sortie la pile sans le sommet, si possible

// val est l'entier qui était au sommet de la pile

// la fonction retourne vrai si val a été dépilée, faux sinon

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 88
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

18.4 SOLUTION : manipuler une pile d'entiers gérée avec un tableau et un indice.(procédure init_pile)

Procédure init_pile (**Sortie** lifo : pile)

// Cette procédure permet d'initialiser une pile, c'est à dire qu'elle n'ait aucun élément

// lifo est la pile initialisée

Début

// il n'y a pas d'élément dans la pile

lifo.indice := 0

Fin

18.5 SOLUTION : manipuler une pile d'entiers gérée avec un tableau et un indice.(procédure empiler)

Fonction empiler (**Entrée Sortie** lifo : pile, **Entrée** val : **entier**) : **booléen**

// Cette fonction permet d'insérer un élément dans la pile.

// Si la pile est pleine, la fonction ne fait rien, mais retourne faux.

// lifo est en entrée la pile dans laquelle on veut empiler

// en sortie la pile dans laquelle val est empilé si possible

// val est la valeur à empiler

// la fonction retourne vrai si val a été empilé, faux sinon

Variables ok : **booléen** // ok est vrai si l'empilage se passe bien

Début

// si la pile est pleine on ne peut pas empiler

Si lifo.indice = taille **Alors**
ok := **faux**

Sinon // on range l'élément dans le tableau
lifo.indice := lifo.indice + 1
lifo.table [lifo.indice] := val
ok := **vrai**

Finsi

// on retourne le compte rendu de l'empilage

Retourner (ok)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 89
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

18.6 SOLUTION : manipuler une pile d'entiers gérée avec un tableau et un indice. (Fonction dépiler)

Fonction dépiler (**Entrée** **Sortie** lifo : pile, **Sortie** val : entier) : booléen

// Cette fonction permet de récupérer l'élément au sommet de la pile.
 // Si la pile est vide, la fonction ne fait rien, mais retourne faux

// lifo est en entrée la pile dans laquelle on veut prendre une valeur
 // en sortie la pile sans le sommet, si possible
 // val est l'entier qui était au sommet de la pile

// la fonction retourne vrai si val a été dépilée, faux sinon

Variables ok : booléen // ok est vrai si le dépilage se passe bien

Début

// si la pile est vide on ne peut pas dépiler

Si lifo.indice = 0 **Alors**

ok := faux

Sinon

// on prend le sommet du tableau

val := lifo.table [lifo.indice]
 lifo.indice := lifo.indice - 1
 ok := vrai

Finsi

// on retourne le compte rendu de l'empilage

Retourner (ok)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 90
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

19. EXERCICE : manipuler une table gérée par Hash-Code

Soit une table contenant des informations, chaque information étant repérée par un identificateur unique. Le problème consiste à obtenir un accès performant à cette information, tout en ayant une souplesse d'utilisation (ajout et retrait d'une information performant).

La dichotomie est une solution possible à ce problème. Cette solution a cependant deux inconvénients :

- Le nombre de recherches à effectuer est de l'ordre de $\log_2(n)$ (n étant le nombre d'éléments de la table).
- L'ajout d'un élément en cours d'exploitation de la table est lourd à gérer.

Voici une autre méthode :

Soit une fonction qui, à chaque nom, fait correspondre un nombre compris entre 1 et la taille de la table. Appelons `h_code` cette fonction.

Cette fonction donne pour chaque nom un accès dans la table d'informations. Hélas deux noms complètement différents pourraient avoir la même entrée dans la table. Cela s'appelle une collision.

La solution proposée ici va donc traiter deux problèmes :

- Choisir une bonne fonction de `hash_code` (`h_code`). Ce n'est pas le problème à traiter ici.
- Gérer les collisions.

Voici la table utilisée :

	nom	information	suivant	
1	Jules	0	Début de zone normale
2	Antoine	0	
2				
4				
5	Mathieu	1001	
.				
.				
.				
1000				Fin de zone normale
1001	Jérôme	0	Début de zone de collision
.				
.				
1050				Fin de zone de collision

Dans la table Mathieu et Jérôme ont le même accès (5) donc le deuxième entré dans la table est géré dans la zone de collision. Si il y avait un troisième nom ayant le même accès, il serait rangé dans la zone collision en étant le suivant de Jérôme (pas forcément le suivant dans la table, mais le suivant au sens des listes).

Recherche de l'indice d'un élément dans la table : La fonction `h_code` donne une entrée dans la table dont le nom correspond à celui que l'on recherche : c'est fini, l'indice est le résultat du `h_code`.

La fonction `h_code` donne une entrée dans la table dont le nom ne correspond pas à celui recherché. La liste des noms ayant le même code d'entrée sera parcourue. Si le nom est trouvé, le résultat est l'indice dans le tableau où se trouve ce nom. Sinon le résultat sera 0, indiquant que le nom n'est pas dans la table.

Ecrivez la procédure indiquant pour un nom donné son emplacement dans la table, ou 0 si il ne s'y trouve pas.

Note sur le `hash_code` : Pour une bonne fonction de `hash_code` (qui est supposée fournie), et avec un taux de remplissage de la table inférieur à 95 % le nombre moyen de comparaisons pour accéder à l'information ne dépasse pas 2. Ce nombre croît très vite avec des taux de remplissage de la table supérieurs. Il est donc recommandé de surdimensionner la table d'environ 10% par rapport à ses besoins.

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 91
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

19.1 SOLUTION : manipuler une table gérée par h_code.(structure de données)

Constantes

```

tailtable      = 1050      // taille du tableau contenant les noms
tailtablenormal = 1000    // taille de la zone à accès direct
tailnom        = 20        // les noms ont dix caractères maximum
final          = 0         // indicateur de fin de liste

```

Types

```

chaînenom = tableau [ tailnom ] de caractère
                                                    // type des noms rangés dans le tableau
information = ...                               // peu importe ici le contenu de l'information

élément = Enregistrement // type des éléments de la table des noms
    nom : chaînenom        // nom est la clef d'accès au contenu dans la table
    info : information      // information est l'ensemble des informations
                            // associées au nom
    suivant : entier       // indice du nom suivant de même clef dans la table
Finenregistrement

table = tableau [ tailtable ] de élément
                                                    // table est la table des noms et des informations
                                                    // associées permettant l'accès par h_code

```

Principe de fonctionnement de la recherche sur une table accédée par hash_code

La fonction de recherche d'un nom, dans une table gérée par hash_code, rend soit l'indice sur l'entrée de la table contenant le nom recherché, soit zéro qui est le code de fin de liste.

Note : On suppose qu'il est possible de comparer des chaînes de caractères de même type entre elles.

19.2 SOLUTION : manipuler une table gérée par h_code.(fonction rechercher_nom)

Fonction chercher_nom (Entrée tabnom : table, Entrée nom : chaînenom) : entier

// Cette fonction cherche une entrée de la table ayant pour clef le nom
// donné. Si le nom n'est pas dans la table elle rend le code "final".

// tabnom est la table gérée par h_code
// nom est le nom à chercher dans le tableau.

// la fonction retourne l'indice de l'entrée de la table ayant le nom pour clef
// 0 = final si le nom n'est pas dans la table

// la constante final a été définie en amont et représente la fin de liste

Variables indice : entier // indice est l'indice de recherche dans la table

Fonction h_code (Entrée nom : chaînenom) : entier

// Cette fonction calcule une clef d'accès à partir du nom donné

// nom est le nom à partir duquel on calcule la clef d'accès

// la fonction retourne un entier compris entre 1 et "tailtablenormal"

Début // calcul de la clef d'accès dans la table

indice := h_code (nom)

// parcours de la liste des noms ayant cette clef d'accès

Tantque (indice <> final) **et** (table [indice].nom <> nom) **Faire**
// arrêt quand on est au bout de la liste, ou quand le nom est trouvé

indice := table [indice].suivant

Fintantque

// soit le nom est trouvé, soit indice vaut 0 = final
// on retour du compte rendu de la recherche

Retourner (indice)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 93
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

20. EXERCICE : gérer une liste de noms classés alphabétiquement.

Il s'agit de constituer une liste de noms classés par ordre alphabétique. Cette liste peut évoluer dans le temps, un nom pouvant être ajouté ou retranché à cette liste, tout en conservant l'ordre alphabétique.

Pour cela, et pour optimiser les temps de traitement, utilisez la structure de données suivante :

```
Constantes    tailtable = 100 // taille du tableau contenant les noms
                tailnom = 10 // les noms ont dix caractères maximum
                final = 0 // indicateur de fin de liste
```

```

Types chaînenom = tableau [ tailnom ] de caractère
                                // type des noms rangés dans le tableau
élément = Enregistrement    // type des éléments de la table des noms
    nom : chaînenom            // nom est le nom contenu dans la table
    suivant : entier          // indice du nom suivant dans la table
Finenregistrement

tablenom = Enregistrement    // type des tables de noms
    table : tableau [ tailtable ] de élément
                                // table permet de ranger les informations
    libre : entier            // libre est le premier élément de la liste libre
    premier : entier          // premier est le premier élément de la liste nom
Finenregistrement

```

Exemple de table :

1	Cunégonde	7
2	Berthe	6
3	Sidonie	0
4		8
5	Agathe	2
6	Cellulite	1
7	Raymonde	3
8		9
9		10

```
premier = 5
libre = 4
```

- 1) donnez une procédure permettant d'initialiser la table des noms à l'origine (vide).
- 2) donnez une procédure permettant d'ajouter un nom à la table des noms.
- 3) donnez une procédure permettant d'enlever un nom de la table des noms.

20.1 SOLUTION : gérer une liste de noms classés alphabétiquement.(structures de données)

```
Constantes    tailtable= 100 // taille du tableau contenant les noms
                tailnom  = 10  // les noms ont dix caractères maximum
                final    = 0    // indicateur de fin de liste
```

Types chaînenom = **tableau** [tailnom] **de caractère**
// type des noms rangés dans le tableau

```
élément      = Enregistrement      // type des éléments de la table des noms
nom  : chaînenom      // nom est le nom contenu dans la table
suivant      : entier      // indice du nom suivant dans la table
```

Finenregistrement

tablenom	= Enregistrement	// type des tables de noms
table	: tableau [tailtable] de élément	// table permet de ranger les informations
libre	: entier	// libre est le premier élément de la liste libre
premier	: entier	// premier est le premier élément de la liste nom
<u>Finenregistrement</u>		

Principe de fonctionnement de la table des noms

La table de noms est vide quand premier vaut « final ».
La table de noms est pleine quand libre vaut « final ».

« ajouter » et « enlever » sont des fonctions qui retournent un entier qui indique si la fonction s'est bien déroulée.

Si on ajoute un nom à une table pleine, la fonction retourne la valeur -1.

Si on enlève un nom à une table vide, la fonction retourne la valeur -1.

Si on ajoute un nom déjà présent dans la table, la fonction retourne la valeur 1.

Si on enlève un nom absent de la table, la fonction retourne la valeur 1.

Dans les autres cas ces deux fonctions retournent des comptes rendus à 0.

Note : Il est supposé qu'il est possible de comparer des chaînes de caractères de même type entre elles.

20.1.1 SOLUTION : gérer une liste de noms classés alphabétiquement.(interfaces)**Procédure** init_table (**Sortie** tabnom : tablenom)

```
// Cette procédure permet d'initialiser une table de noms, c'est à dire
// qu'elle n'ait aucun élément
// tabnom est la table de noms initialisée
// la constante final a été définie en amont et représente la fin de liste
```

Fonction ajouter (**Entrée Sortie** tabnom : tablenom, **Entrée** nom : chaînenom) : **entier**

```
// Cette fonction permet d'ajouter un nom dans la table des noms.
// Si la table de noms est pleine la fonction ne fait rien, mais retourne -1
// Si le nom est déjà présent dans la table de noms, la fonction ne fait rien et retourne la valeur 1

// tabnom est en entrée la table de noms dans laquelle on veut ajouter
// en sortie la table de nom dans laquelle nom a pris sa place si possible.
// nom est le nom à ranger dans le tableau.

// la fonction retourne -1 si la table est pleine
// 0 si le nom est rangé dans la table
// 1 si le nom était déjà dans la table
// la constante final a été définie en amont et représente la fin de liste
```

Fonction enlever (**Entrée Sortie** tabnom : tablenom, **Entrée** nom : chaînenom) : **entier**

```
// Cette fonction permet d'enlever un nom de la table des noms.
// Si la table de noms est vide la fonction ne fait rien, mais retourne -1.
// Si le nom est absent de la table de noms, la fonction ne fait rien, mais retourne la valeur 1.

// tabnom est en entrée la table de noms dans laquelle on veut enlever
// en sortie la table de nom dans laquelle nom a été enlevé si possible.
// nom est le nom à enlever du tableau.

// la fonction retourne -1 si la table est vide
// 0 si le nom est enlevé de la table
// 1 si le nom n'était pas dans la table
```

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 96
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

20.2 SOLUTION : gérer une liste de noms classés alphabétiquement.(procédure init_table)

Procédure init_table (**Sortie** tabnom : tablenom)

// Cette procédure permet d'initialiser une table de noms, c'est à dire
 // qu'elle n'ait aucun élément
 // tabnom est la table de noms initialisée
 // la constante final a été définie en amont et représente la fin de liste

Début // il n'y a pas d'élément dans la table des noms

tabnom.premier := final

// construction de la liste des libres

tabnom.libre := 1

Tantque tabnom.libre <> tailtable **Faire** // création de la liste des « tailtable » éléments libres
 tabnom.table [tabnom.libre].suivant := tabnom.libre + 1
 tabnom.libre := tabnom.libre + 1

Fintantque

// insertion de la fin de la liste des libres

tabnom.table [tabnom.libre].suivant := final

// initialisation au début de la liste des libres

tabnom.libre := 1

Fin

dfpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 97
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

20.3 SOLUTION : gérer une liste de noms classés alphabétiquement.(fonction ajouter)

Fonction ajouter (**Entrée Sortie** tabnom : tablenom, **Entrée** nom : chaînenom) : **entier**

```
// Cette fonction permet d'ajouter un nom dans la table des noms.
// Si la table de noms est pleine la fonction ne fait rien, mais retourne -1
// Si le nom est déjà présent dans la table de noms, la fonction ne fait rien et retourne la valeur 1

// tabnom est en entrée la table de noms dans laquelle on veut ajouter
// en sortie la table de nom dans laquelle nom a pris sa place si possible.
// nom est le nom à ranger dans le tableau.

// la fonction retourne -1 si la table est pleine
// 0 si le nom est rangé dans la table
// 1 si le nom était déjà dans la table
// la constante final a été définie en amont et représente la fin de liste
```

Variables

```
ok      : entier // ok est le code de retour de la fonction
précède : entier // précède est le nom avant le nom inspecté
ptnom   : entier // ptnom est l'indice du nom inspecté
nouvel  : entier // nouvel est l'indice ou on va ranger le nom
```

Début

Si tabnom.libre = final **Alors** // si la table de noms est pleine, rajout impossible

```
ok := -1
```

Sinon // recherche du lieu de rangement du nom dans le tableau

```
précède := final
ptnom := tabnom.premier
```

```
// parcours de la liste des noms jusqu'à trouver le nom à l'indice ptnom
// sa place entre précède et ptnom, ou la fin de liste
```

Tantque (ptnom <> final) **et** (tabnom.table [ptnom].nom < nom) **Faire**
 // arrêt en fin de liste, ou quand on trouve
 // un nom plus grand ou égal dans la table

```
précède := ptnom
ptnom := tabnom.table [ ptnom ].suivant
```

Fintantque

```
// soit on a trouvé le nom, soit il faut ajouter le nom en tête, soit il faut
// ajouter le nom en milieu ou fin de liste.
```

Si (ptnom <> final) **et** (nom = tabnom.table [ptnom].nom) **Alors**

```
// nom trouvé dans la table des noms
```

```
ok := 1
```

Sinon // rangement du nom dans la table des noms

```
ok := 0
nouvel := tabnom.libre
```

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 98
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

```

tabnom.table [ nouvel ].nom := nom

// remise à jour la liste des livres

tabnom.libre := tabnom.table [ tabnom.libre ].suivant

// soit on insère le nom en tête de liste, soit on l'insère dans la liste

Si précède = final Alors
    // soit c'est le premier élément de la table de noms
    // soit nom est le plus petit nom de la table
    tabnom.table [ nouvel ].suivant := tabnom.premier
    tabnom.premier := nouvel
Sinon
    // soit on insère au milieu de la liste de noms
    // soit en fin de liste de noms
    tabnom.table [ précède ].suivant := nouvel
    tabnom.table [ nouvel ].suivant := ptnom
Finsi

Finsi

Finsi

// retour du compte rendu de l'ajout

Retourner ( ok )

Fin

```

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 99
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

20.4 SOLUTION : gérer une liste de noms classés alphabétiquement.(fonction enlever)

Fonction enlever (**Entrée Sortie** tabnom : tablenom, **Entrée** nom : chaînenom) : **entier**

// Cette fonction permet d'enlever un nom de la table des noms.
 // Si la table de noms est vide la fonction ne fait rien, mais retourne -1.
 // Si le nom est absent de la table de noms, la fonction ne fait rien, mais retourne la valeur 1.

 // tabnom est en entrée la table de noms dans laquelle on veut enlever
 // en sortie la table de nom dans laquelle nom a été enlevé si possible.
 // nom est le nom à enlever du tableau.

 // la fonction retourne -1 si la table est vide
 // 0 si le nom est enlevé de la table
 // 1 si le nom n'était pas dans la table

Variables ok : **entier** // ok est le code de retour de la fonction
 précède: **entier** // précède est le nom avant le nom inspecté
 ptnom : **entier** // ptnom est l'indice du nom inspecté

Début

// si la table de noms est vide on ne peut pas enlever

Si tabnom.premier = final **Alors**

ok := -1

Sinon

// recherche du nom dans le tableau

précède := final
 ptnom := tabnom.premier

// parcours de la liste des noms jusqu'à trouver le nom à l'indice ptnom,
 // sa place entre précède et ptnom, ou la fin de liste

Tantque (ptnom <> final) **et** (tabnom.table [ptnom].nom < nom) **Faire**

// arrêt en fin de liste, ou quand on trouve
 // un nom plus grand ou égal dans la table

précède := ptnom
 ptnom := tabnom.table [ptnom].suivant

Fintantque

// soit on a trouvé le nom, et on a le précédent, soit on ne l'a pas trouvé

Si (ptnom = final) **ou** (nom <> tabnom.table [ptnom].nom) **Alors**

// nom pas trouvé dans la table des noms

ok := 1

Sinon

// enlever le nom de la table des noms

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 100
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

ok := 0

// soit on enlève le nom en tête de liste, soit on l'enlève dans la liste

Si précède = final **Alors** // c'est le premier élément de la table de noms

tabnom.premier := tabnom.table [ptnom].suivant

Sinon // soit on enlève dans la liste de noms

tabnom.table [précède].suivant := tabnom.table [ptnom].suivant

Finsi

// il faut raccrocher ptnom à la liste des libres

tabnom.table [ptnom].suivant := tabnom.libre

tabnom.libre := ptnom

Finsi

Finsi

// retour du compte rendu du retrait

Retourner (ok)

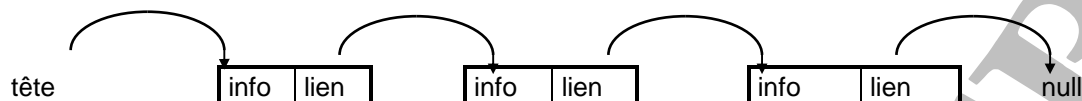
Fin

dipa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 101
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

21. EXERCICE : créer une liste dynamique gérée en FIFO.

Une liste dynamique est une liste à laquelle on peut rajouter un nombre quelconque d'éléments sans restriction de taille (sauf celle de la mémoire). Les éléments de la liste sont chaînés entre eux.

Structure d'une telle liste :



Structure d'un élément de la liste:

type

référence = **pointeur de** élément

élément = **enregistrement**

valeur : info

// contient l'information que l'on désire conserver

suivant : référence

// pointeur vers l'élément suivant de la liste

finenregistrement

La liste sera repérée par un pointeur sur son premier élément (la tête de la liste).

Elle se termine par le premier élément trouvé dont le lien est à **null** ou avec le pointeur de tête à **null** (en cas de liste vide).

Soit une liste dynamique d'entiers fonctionnant en FIFO (First In First Out). Le dernier élément de la liste est le dernier élément rentré dans la liste. Le premier élément de la liste est le premier élément rentré dans la liste et le premier élément qui sortira de la liste.

Ecrire une procédure qui rajoute un entier à la liste.

Ecrire une procédure qui récupère un élément de la liste.

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 102
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

21.1 SOLUTION : créer une liste dynamique gérée en FIFO.(données)

Type pt_elem = pointeur de élément // type des pointeur sur des chaînon de la liste

élément = Enregistrement // chaînon de la liste

valeur : entier // valeur conservée dans ce chaînon

suivant : pt_elem // pointeur vers le chaînon suivant de la liste

Finenregistrement

Principe de fonctionnement de la liste dynamique

Il n'y a pas de contraintes pour ajouter un élément à la liste.

Par contre, il n'est pas possible d'enlever un élément d'une liste qui serait vide.

Il est judicieux, par souci de cohérence, que ajouter et enlever soient deux fonctions.

La première rendra un indicateur vrai, la seconde rendra un indicateur vrai chaque fois que la liste en entrée ne sera pas vide, faux sinon.

21.2 SOLUTION : créer une liste dynamique gérée en FIFO.(interfaces)

Procédure initliste (Sortie tête : pt_elem)

// Cette procédure permet d'initialiser une liste, c'est à dire sans aucun élément

// tête est la tête de la liste créée

Fonction ajouter (Entrée Sortie tête : pt_elem, Entrée val : entier) : booléen

// Cette fonction permet d'insérer un élément à la fin de la liste

// tête est en entrée la tête de la liste

// en sortie la tête de la liste, modifiée si tête était égal à null

// val est la valeur à ajouter en fin de liste

// la fonction retourne toujours vrai

Fonction enlever (Entrée Sortie tête : pt_elem, Sortie val : entier) : booléen

// Cette fonction permet de récupérer l'élément en tête de la liste.

// Si la liste est vide, la fonction ne fait rien, mais retourne faux

// tête est en entrée la tête de la liste en sortie la tête de la nouvelle liste

// val sera l'entier qui était en tête de la liste

// la fonction retourne vrai si val a été enlevé, faux si la liste était vide

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 103
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

21.3 SOLUTION : créer une liste dynamique gérée en FIFO.

Algorithmes des procédures et fonctions

Procédure initliste (**Sortie** tête : pt_elem)

// Cette procédure permet d'initialiser une liste, c'est à dire sans aucun élément
// tête est la tête de la liste créée

Début

// il n'y a pas d'élément dans la liste

tête := **null**

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 104
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

21.4 SOLUTION : créer une liste dynamique gérée en FIFO.(fonction ajouter)

Fonction ajouter (Entrée Sortie tête : pt_elem, Entrée val : entier) : booléen

// Cette fonction permet d'insérer un élément à la fin de la liste
 // tête est en entrée la tête de la liste
 // en sortie la tête de la liste, modifiée si tête était égal à null
 // val est la valeur à ajouter en fin de liste

 // la fonction retourne toujours vrai

Variables pt_parcours : pt_elem // pt_parcours permet de parcourir la liste

Début

Si tête = null **Alors**

// la liste est vide on ajoute en tête

créer (tête)
 tête->.valeur := val
 tête->.suivant := null

Sinon // la liste n'est pas vide, ajout en queue

pt_parcours := tête
Tantque pt_parcours->.suivant <> null **Faire**
 // arrêt sur le dernier chaînon de la liste

pt_parcours := pt_parcours->.suivant

Fintantque // on raccroche en queue le nouveau chaînon

créer (pt_parcours->.suivant)
 pt_parcours->.suivant->.valeur := val
 pt_parcours->.suivant->.suivant := null

Finsi

// on retourne le compte rendu de l'ajout

Retourner (vrai)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 105
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

21.5 SOLUTION : créer une liste dynamique gérée en FIFO.(fonction enlever)

Fonction enlever (**Entrée Sortie** tête : pt_elem, **Sortie** val : entier) : **booléen**

// Cette fonction permet de récupérer l'élément en tête de la liste.

// Si la liste est vide, la fonction ne fait rien, mais retourne faux

// tête est en entrée la tête de la liste en sortie la tête de la nouvelle liste

// val sera l'entier qui était en tête de la liste

// la fonction retourne vrai si val a été enlevé, faux si la liste était vide

Variables ok : **booléen** // ok est vrai si l'enlèvement se passe bien
pt_dét : pt_elem // pt_det sert à mémoriser l'élément à détruire

Début

Si tête = **null** **Alors**

// la liste est vide on ne peut pas enlever un élément

ok := **faux**

Sinon // on prend le premier élément de la liste

val := tête ->.valeur

pt_det := tête

tête := tête ->.suivant

// l'élément est ôté de la liste

détruire (pt_det)

// l'élément est détruit car inutile

ok := **vrai**

Finsi

// retour du le compte-rendu de l'enlèvement d'un élément

Retourner (ok)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 106
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

22. EXERCICE : gérer une liste dynamiquement.

Il s'agit de faire le parallèle entre l'utilisation des pointeurs et les utilisations des indices pour désigner le nom suivant de la liste dans l'exercice des noms classés alphabétiquement.

Refaites l'exercice des noms classés alphabétiquement à l'aide d'une liste dynamique.

22.1 SOLUTION : gérer une liste dynamiquement.(structures de données)

Constantes tailnom = 10 // les noms ont dix caractères maximum

```

Types chaînenom      = tableau [ tailnom ] de caractère
                                // type des noms rangés dans la liste de noms
pt_elem      = pointeur de élément
                                // type des pointeur sur des chaînons de la liste
élément      = Enregistrement      // chaînon de la liste
    nom : chaînenom      // nom conservé dans ce chaînon
    suivant : pt_elem     // pointeur vers le chaînon suivant de la liste
Finenregistrement

```

Principe de fonctionnement de la table des noms

Il n'y a pas de contrainte de débordement quand un élément est ajouté.

Par contre, quand un élément d'une liste vide doit être enlevé, il faut signaler une erreur.

« ajouter » et « enlever » sont des fonctions qui retournent un entier qui indique si la fonction s'est bien déroulée.

Si un nom à une liste vide est enlevé, la fonction retourne la valeur -1.

Si un nom déjà présent dans la liste est ajouté, la fonction retourne la valeur 1.

Si un nom absent de la liste est enlevé, la fonction retourne la valeur 1.

Dans les autres cas, ces deux fonctions retourneront des comptes rendus à 0.

Note : Il est supposé qu'il est possible de comparer des chaînes de caractères de même type entre elles.

22.2 SOLUTION : gérer une liste dynamiquement.(interfaces)

Procédure init_liste (**Sortie** tête : pt_elem)

// Cette procédure permet d'initialiser une liste, c'est à dire qu'elle n'ait aucun élément

// tête est la tête de la liste créée

Fonction ajouter (**Entrée Sortie** tête : pt_elem, **Entrée** nom : chaînenom) : **entier**

// Cette fonction permet d'ajouter un nom dans la liste des noms.

// Si le nom est déjà présent dans la liste de noms, la fonction ne fait rien,

// mais retourne la valeur 1.

// tête est en entrée la tête de liste de noms dans laquelle on veut ajouter un nom

// en sortie la tête de liste de noms dans laquelle nom a pris sa place si possible

// nom est le nom à ranger dans la liste.

// la fonction retourne 0 si le nom est rangé dans la liste

// 1 si le nom était déjà dans la liste

Fonction enlever (**Entrée Sortie** tête : pt_elem, **Entrée** nom : chaînenom) : **entier**

// Cette fonction permet d'enlever un nom de la liste des noms.

// Si la liste de noms est vide la fonction ne fait rien, mais retourne -1.

// Si le nom est absent de la liste de noms, la fonction ne fait rien,

// mais retourne la valeur 1.

// tête est en entrée la tête de liste de noms dans laquelle on veut enlever un nom

// en sortie la tête de liste de nom dans laquelle nom a été enlevé si possible.

// nom est le nom à enlever de la liste.

// la fonction retourne -1 si la liste était vide

// 0 si le nom est enlevé de la liste

// 1 si le nom n'était pas dans la liste

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 108
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

22.3 SOLUTION : gérer une liste dynamiquement.(procédure init_liste)

Procédure init_liste (**Sortie** tête : pt_elem)

// Cette procédure permet d'initialiser une liste, c'est à dire qu'elle n'ait aucun élément

// tête est la tête de la liste créée

Début

// il n'y a pas d'élément dans la liste

tête := **null**

Fin

afpa©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 109
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

22.4 SOLUTION : gérer une liste dynamiquement.(fonction ajouter)

Fonction ajouter (**Entrée** **Sortie** tête : pt_elem, **Entrée** nom : chaînenom) : **entier**

```
// Cette fonction permet d'ajouter un nom dans la liste des noms.
// Si le nom est déjà présent dans la liste de noms, la fonction ne fait rien,
// mais retourne la valeur 1.

// tête est en entrée la tête de liste de noms dans laquelle on veut ajouter un nom
// en sortie la tête de liste de noms dans laquelle nom a pris sa place si possible
// nom est le nom à ranger dans la liste.

// la fonction retourne 0 si le nom est rangé dans la liste
// 1 si le nom était déjà dans la liste
```

Variables ok : **entier** // ok est le code de retour de la fonction
 pt_prec : pt_elem // pointeur sur le chaînon précédent celui en cours
 pt_cour : pt_elem // pointeur sur le chaînon courant de la liste

Début // initialisation des pointeurs de parcours de la liste à la recherche du nom

```
pt_prec := null
pt_cour := tête
```

```
// parcours de la liste des noms jusqu'à trouver le chaînon où est le nom,
// les deux chaînons entre lesquels il devrait être, ou la fin de liste
```

Tantque (pt_cour <> **null**) **et** (ptcour->.nom < nom) **Faire**
 // arrêt en fin de liste, ou quand on trouve un nom plus grand ou égal dans la liste

```
pt_prec := pt_cour
pt_cour := pt_cour->.suivant
```

Fintantque

```
// soit on a trouvé le nom, soit il faut ajouter le nom en tête,
// soit il faut ajouter le nom en milieu ou fin de liste.
```

Si (pt_cour <> **null**) **et** (nom = pt_cour->.nom) **Alors**
 ok := 1 // le nom est trouvé dans la liste des noms

Sinon // rangement du nom dans la liste alphabétique des noms
 ok := 0

Si pt_prec = **null** **Alors** // soit c'est le premier élément de la liste de noms
 // soit nom est le plus petit nom de la liste

```
créer ( tête )
tête->.nom := nom
tête->.suivant := pt_cour
```

Sinon // soit on insère au milieu de la liste de noms, soit en fin de liste de noms

```
créer ( pt_prec->.suivant )
pt_prec->.suivant->.nom := nom
pt_prec->.suivant->.suivant := pt_cour
```

Finsi

Finsi

```
// retour du compte rendu de l'ajout
```

Retourner (ok)

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 110
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

Fin

Document AFPA

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 111
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

22.5 SOLUTION : gérer une liste dynamiquement.(fonction enlever)

Fonction enlever (**Entrée Sortie** tête : pt_elem, **Entrée** nom : chaînenom) : **entier**

// Cette fonction permet d'enlever un nom de la liste des noms.
 // Si la liste de noms est vide la fonction ne fait rien, mais retourne -1.
 // Si le nom est absent de la liste de noms, la fonction ne fait rien,
 // mais retourne la valeur 1.

// tête est en entrée la tête de liste de noms dans laquelle on veut enlever un nom
 // en sortie la tête de liste de nom dans laquelle nom a été enlevé si possible.
 // nom est le nom à enlever de la liste.

// la fonction retourne -1 si la liste était vide
 // 0 si le nom est enlevé de la liste
 // 1 si le nom n'était pas dans la liste

Variables ok : **entier** // ok est le code de retour de la fonction
 pt_prec : pt_elem // pointeur sur le chaînon précédent celui en cours
 pt_cour : pt_elem // pointeur sur le chaînon courant de la liste

Début // si la liste des noms est vide on ne peut pas enlever

Si tête = **null** **Alors**
 ok := -1

Sinon // recherche du nom dans la liste

pt_prec := **null**
 pt_cour := tête

// parcours de la liste des noms jusqu'à trouver le chaînon où est le nom,
 // les deux chaînons entre lesquels il devrait être, ou la fin de liste

Tantque (pt_cour <> **null**) **et** (pt_cour->.nom < nom) **Faire**
 // arrêt en fin de liste, ou quand on trouve un nom plus grand ou égal dans la liste

pt_prec := pt_cour
 pt_cour := pt_cour->.suivant

Fintantque // soit on a trouvé le nom, soit on ne l'a pas trouvé et on arrête là

Si (pt_cour = **null**) **ou** (nom <> pt_cour->.nom) **Alors**
 ok := 1 // nom non trouvé dans la liste des noms

Sinon // nom enlevé de la liste des noms
 ok := 0 // soit on enlève le nom en tête de liste, soit on l'enlève dans la liste

Si pt_prec = **null** **Alors** // c'est le premier élément de la liste de noms
 tête := tête->.suivant

Sinon // soit on enlève dans la liste de noms
 pt_prec->.suivant := pt_cour->.suivant

Finsi

détruire (pt_cour) // chaînon contenant le nom à détruire

Finsi

Finsi // retour du compte rendu du retrait

Retourner (ok)

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 112
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

23. EXERCICE : fichier séquentiel de nombres conservés par ordre croissant.

Ecrire une procédure qui permet d'ajouter un nombre à un fichier de nombres classé par ordre croissant, le fichier résultant étant lui aussi classé par ordre croissant.

23.1 SOLUTION : fichier séquentiel croissant.(Programme)

Programme fichier_séquentiel_croissant

// ce programme constitue et complète des fichiers de nombres croissants

Constantes taille = 16 // taille maximale d'un nom de fichier

Types nomfichier = tableau [taille] de caractères
// type des noms de fichier

Variables nomfic : nomfichier // nom du fichier à compléter
nombre : entier // nombre à entrer dans le fichier
fic : fichier de entier // fichier de nombres classés

Procédure ajouter (Entrée nomfiche : nomfichier, Entrée nb : entier)

// cette procédure ajoute un nombre à un fichier classé par ordre croissant,
// le fichier résultant étant lui-même classé par ordre croissant.

// nomfiche est le nom du fichier
// nb est le nombre à classer dans le fichier

Début

// saisie du nom du fichier et du nombre à classer

écrire ('donnez le nom du fichier : ')

lire (nomfic)

écrire ('donnez le nombre à insérer : ')

lire (nombre)

// appel de la procédure d'ajout d'un nombre dans le fichier

ajouter (nomfic, nombre)

// affichage du fichier classé par ordre croissant

ouvrir (nomfic, fic) // ouverture du fichier

Tantque non finfichier (fic) **Faire**
// arrêt en fin de fichier

lire (fic, nombre) // lecture d'un nombre du fichier

écrire (nombre) // affichage du nombre

Fintantque

fermer (fic) // fermeture du fichier

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 113
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

23.2 SOLUTION : fichier séquentiel croissant. (Jeux d'essais)

23.2.1 essai avec un seul fichier

nomfic = 'toto.nb'	nombre = 25
nomfic = 'toto.nb'	nombre = 12
nomfic = 'toto.nb'	nombre = 52
nomfic = 'toto.nb'	nombre = 47
nomfic = 'toto.nb'	nombre = 17
nomfic = 'toto.nb'	nombre = 2
nomfic = 'toto.nb'	nombre = 8795

23.2.2 essai avec plusieurs fichiers

nomfic = 'tata.nb'	nombre = 12
nomfic = 'tutu.nb'	nombre = 52
nomfic = 'tyty.nb'	nombre = 8795
nomfic = 'tyty.nb'	nombre = 2
nomfic = 'tutu.nb'	nombre = 47
nomfic = 'tata.nb'	nombre = 17
nomfic = 'tutu.nb'	nombre = 25

23.3 SOLUTION : fichier séquentiel croissant. (Procédure ajouter)

Procédure ajouter (**Entrée** nomfiche : nomfichier, **Entrée** nb : **entier**)

// cette procédure ajoute un nombre à un fichier classé par ordre croissant,
 // le fichier résultant étant lui-même classé par ordre croissant.

// nomfiche est le nom du fichier
 // nb est le nombre à classer dans le fichier

Variables fic : **fichier de entier** // fichier de nombres classés
 fictmp : **fichier de entier** // fichier intermédiaire
 valeur : **entier** // valeur lue dans le fichier initial

Début

// recopie du fichier dans un autre fichier avec insertion du nombre

ouvrir (nomfiche, fic) // ouverture du fichier initial
ouvrir ('fic.tmp', fictmp) // ouverture du fichier intermédiaire

Si **finfichier** (fic) **Alors** // le fichier est vide

écrire (fictmp, nb) // on range la valeur dans le fichier intermédiaire

Sinon // il y a quelque chose dans le fichier

lire (fic, valeur) // lecture d'un nombre du fichier initial

Tantque nb > valeur **et non finfichier** (fic) **faire**
 // arrêt en fin de fichier ou quand le nombre du fichier est plus grand

écrire (fictmp, valeur)

lire (fic, valeur) // lecture d'un nombre du fichier initial

Fintantque

Si nb > valeur **Alors** // recopier la valeur puis le nombre
 // le fichier est vide

écrire (fictmp, valeur)

écrire (fictmp, nb)

Sinon // recopier le nombre puis la valeur
 // si le fichier n'est pas vide on recopiera la fin

écrire (fictmp, nb)

écrire (fictmp, valeur)

Finsi

 // recopie de la fin du fichier

Tantque non finfichier (fic) **faire** // arrêt en fin de fichier

lire (fic, valeur) // lecture d'un nombre du fichier initial

écrire (fictmp, valeur)

Fintantque**Finsi**

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 115
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

fermer (fic) // fermeture du fichier initial
fermer (fictmp) // fermeture du fichier intermédiaire

// recopie du fichier intermédiaire dans le fichier initial

ouvrir (nomfiche, fic) // ouverture du fichier initial
ouvrir ('fic.tmp', fictmp) // ouverture du fichier intermédiaire

Tantque non finfichier (fictmp) **Faire**
 // arrêt en fin de fichier
 lire (fictmp, valeur) // lecture d'un nombre du fichier intermédiaire
 écrire (fic, valeur) // recopie dans le fichier initial

Fintantque

fermer (fic) // fermeture du fichier initial
fermer (fictmp) // fermeture du fichier intermédiaire

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 116
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

24. EXERCICE : gestion d'un fichier de salariés.

Soit un fichier de salariés, où chaque fiche comporte un matricule (unique pour chaque salarié), un nom, et un salaire. Ecrire l'algorithme d'un programme qui permet de créer une nouvelle fiche, de détruire une fiche, de voir une fiche, et de lister l'ensemble des fiches, par numéro de fiche croissant. Le fonctionnement se fera en mode rouleau, sans souci de présentation de l'écran.

24.1 SOLUTION : gestion d'un fichier de salariés.(Programme)**Programme** gérer_fiches_salariés

// ce programme permet de gérer des fichiers de salariés. Nous pouvons créer une fiche
// salarié, la détruire, la voir, et lister l'ensemble des fiches.

Constantes

taille = 16 // taille maximale d'un nom de fichier
long = 20 // longueur maximale d'un nom

Types

nomfichier = **tableau** [taille] **de caractères**
// type des noms de fichier
nomsalarié = **tableau** [long] **de caractères**
// type des noms des salariés
salarié = **enregistrement**
// type des fiches de salarié
matricule : **entier** // matricule du salarié
nom : nomsalarié // nom du salarié
salaire : **réel** // émoluments du salarié
finenregistrement

Variables

nomfic : nomfichier // nom du fichier à compléter
réponse : **caractère** // commande de l'opérateur

Procédure lister (nomfiche : nomfichier)

// cette procédure permet de lister les fiches des salariés contenues dans le fichier
// par ordre croissant de matricule

// nomfiche est le nom du fichier de salarié à traiter

Procédure créer (nomfiche : nomfichier)

// cette procédure permet de créer une fiche de salarié et de l'intégrer dans le fichier des
// salariés

// nomfiche est le nom du fichier de salarié à traiter

Procédure détruire (nomfiche : nomfichier)

// cette procédure permet de détruire une fiche de salarié du fichier des salariés

// nomfiche est le nom du fichier de salarié à traiter

Procédure voir (nomfiche : nomfichier)

// cette procédure permet de voir une fiche de salarié contenue dans le fichier des salariés

// nomfiche est le nom du fichier de salarié à traiter

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 117
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

Début

```
// saisie du nom du fichier
```

écrire ('donnez le nom du fichier : ')

lire (nomfic)

Répéter // jusqu'à avoir la commande de fin

écrire ('l)nsérer, (D)étruire, (V)oir, (L)ister : ')

lire (réponse)

Choix sur réponse Faire

'I' : insérer (nomfic)

'D' : détruire (nomfic)

'V' : voir (nomfic)

'L' : lister (nomfic)

'Q' : // ne rien faire

autres : écrire ('erreur de saisie recommencez.')

Finchoix

jusqua réponse = 'Q' // commande de fin de travail

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 118
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

24.2 SOLUTION : gestion d'un fichier de salariés.(Procédure créer)

Procédure créer (nomfiche : nomfichier)

// cette procédure permet de créer une fiche de salarié et de l'intégrer dans le fichier des
// salariés

// nomfiche est le nom du fichier de salarié à traiter

Variables

fic : **fichier de** salarié // fichier des salariés indexés par le matricule
personne : salarié // fiche d'un salarié

Début

// saisie de la fiche du salarié

écrire ('donnez le matricule du salarié : ')

lire (personne.matricule)

écrire ('donnez le nom du salarié : ')

lire (personne.nom)

écrire ('donnez le salaire du salarié : ')

lire (personne.salaire)

ouvrir (nomfiche, fic, **indexé**) // ouverture du fichier des salariés

// le salarié existe-t'il déjà ?

Si positionner (fic, personne.matricule) **Alors**

// le salarié existe déjà

écrire ('cette fiche a déjà été créée')

Sinon

// on peut entrer la fiche

écrire (fic, personne)

Finsi

fermer (fic) // fermeture du fichier des salariés

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 119
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

24.3 SOLUTION : gestion d'un fichier de salariés.(Procédure détruire)

Procédure détruire (nomfiche : nomfichier)

// cette procédure permet de détruire une fiche de salarié du fichier des salariés

// nomfiche est le nom du fichier de salarié à traiter

Variables fic : **fichier de** salarié // fichier des salariés indexés par le matricule
 personne : salarié // fiche d'un salarié

Début

// saisie du matricule du salarié

écrire ('donnez le matricule du salarié : ')**lire** (personne.matricule)**ouvrir** (nomfiche, fic, **indexé**) // ouverture du fichier des salariés

// le salarié existe-t'il ?

Si **positionner** (fic, personne.matricule) **Alors**

// le salarié existe

détruire (fic) // on l'enlève du fichier**écrire** ('la fiche a été détruite')**Sinon**

// le salarié n'existe pas

écrire ('cette fiche n'existe pas')**Finsi****fermer** (fic) // fermeture du fichier des salariés**Fin**

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 120
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC

24.4 SOLUTION : gestion d'un fichier de salariés.(Procédure voir)

// nomfiche est le nom du fichier de salarié à traiter

Procédure voir (nomfiche : nomfichier)

// cette procédure permet de voir une fiche de salarié contenue dans le fichier des salariés

// nomfiche est le nom du fichier de salarié à traiter

Variables fic : **fichier de** salarié // fichier des salariés indexés par le matricule
 personne : salarié // fiche d'un salarié

Début

// saisie du matricule du salarié

écrire ('donnez le matricule du salarié : ')**lire** (personne.matricule)**ouvrir** (nomfiche, fic, **indexé**) // ouverture du fichier des salariés

// le salarié existe-t'il ?

Si non positionner (fic, personne.matricule) **Alors**

// le salarié n'existe pas

écrire ('cette fiche n'existe pas')**Sinon**

// on peut visualiser la fiche

lire (fic, personne)

// visualisation de la fiche

écrire ('Matricule : ', personne.matricule, 'Nom : ', personne.nom,
 'Salaire : ', personne.salaire)**Finsi****fermer** (fic) // fermeture du fichier des salariés**Fin**

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 121
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	03/06 - v1.0	ALGOPC.DOC

24.5 SOLUTION : *gestion d'un fichier de salariés.(Procédure lister)*

Procédure lister (nomfiche : nomfichier)

// cette procédure permet de lister les fiches des salariés contenues dans le fichier
// par ordre croissant de matricule

```
// nomfiche est le nom du fichier de salarié à traiter
```

Variables fic : **fichier** de salarié // fichier des salariés indexés par le matricule
 personne : salarié // fiche d'un salarié

Début

ouvrir (nomfiche, fic) // ouverture du fichier des salariés

écrire ('liste des salariés')

Tantque **non** **finfichier** (fic) **Faire**
 // arrêt en fin de fichier
 // parcours séquentiel du fichier indexé

lire (fic, personne) // lecture d'un salarié du fichier

```

écrire ( 'Matricule : ', personne.matricule, // affichage du salarié
        'Nom : ', personne.nom,
        'Salaire : ', personne.salaire )

```

Fintantque

écrire ('fin de liste')

fermer (fic) // fermeture du fichier des salariés

Fin

afpa ©	auteur	centre	lecteur	formation	module	séq/item	type doc	millésime	page 122
	JC Corre	Grenoble Nancy	B. Manet	APII	1		prop. corrigé	12/95 - v1.0	ALGOPC.DOC