Cours d'algorithmique

I. Définitions

a. Définition d'un algorithme :

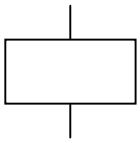
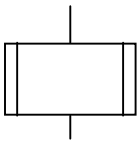
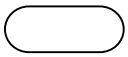
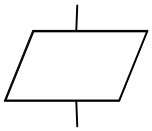
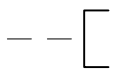
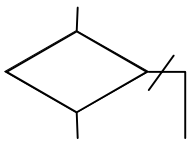
C'est un ensemble de règles opératoires rigoureuses, ordonnant à un processeur d'exécuter dans un ordre déterminé un nombre fini d'opérations élémentaires ; il oblige à une programmation structurée.

Un algorithme est écrit en utilisant un langage de description d'algorithme (LDA).
L'algorithme ne doit pas être confondu avec le programme proprement dit.

b. Définition d'un algorigramme :

C'est une représentation graphique de l'algorithme. Pour le construire, on utilise des symboles normalisés.

Quelques symboles utilisés dans la construction d'un algorigramme :

SYMBOLE	DESIGNATION	SYMBOLE	DESIGNATION
Symboles de traitement		Symboles auxiliaires	
	**Symbole général** Opération ou groupe d'opérations sur des données, instructions, pour laquelle il n'existe aucun symbole normalisé.		**Renvoi** Symbole utilisé deux fois pour assurer la continuité lorsqu'une partie de ligne de liaison n'est pas représentée.
	**Sous-programme** Portion de programme considérée comme une simple opération.		**Début, fin , interruption** Début, fin ou interruption d'un algorigramme.
	**Entrée-Sortie** Mise à disposition d'une information à traiter ou enregistrement d'une information traitée.		**Commentaire** Symbole utilisé pour donner des indications sur les opérations effectuées.
Symbole de test		Les différents symboles sont reliés entre eux par des lignes de liaisons.	
	**Branchement** Exploitation de conditions variables impliquant un choix parmi plusieurs.		
Sens conventionnel des liaisons			
Le sens général des lignes de liaison doit être : • De haut en bas • De gauche à droite Lorsque le sens général ne peut pas être respecté, des pointes de flèche à cheval sur la ligne indiquent le sens utilisé.			

II. L' algorithme :

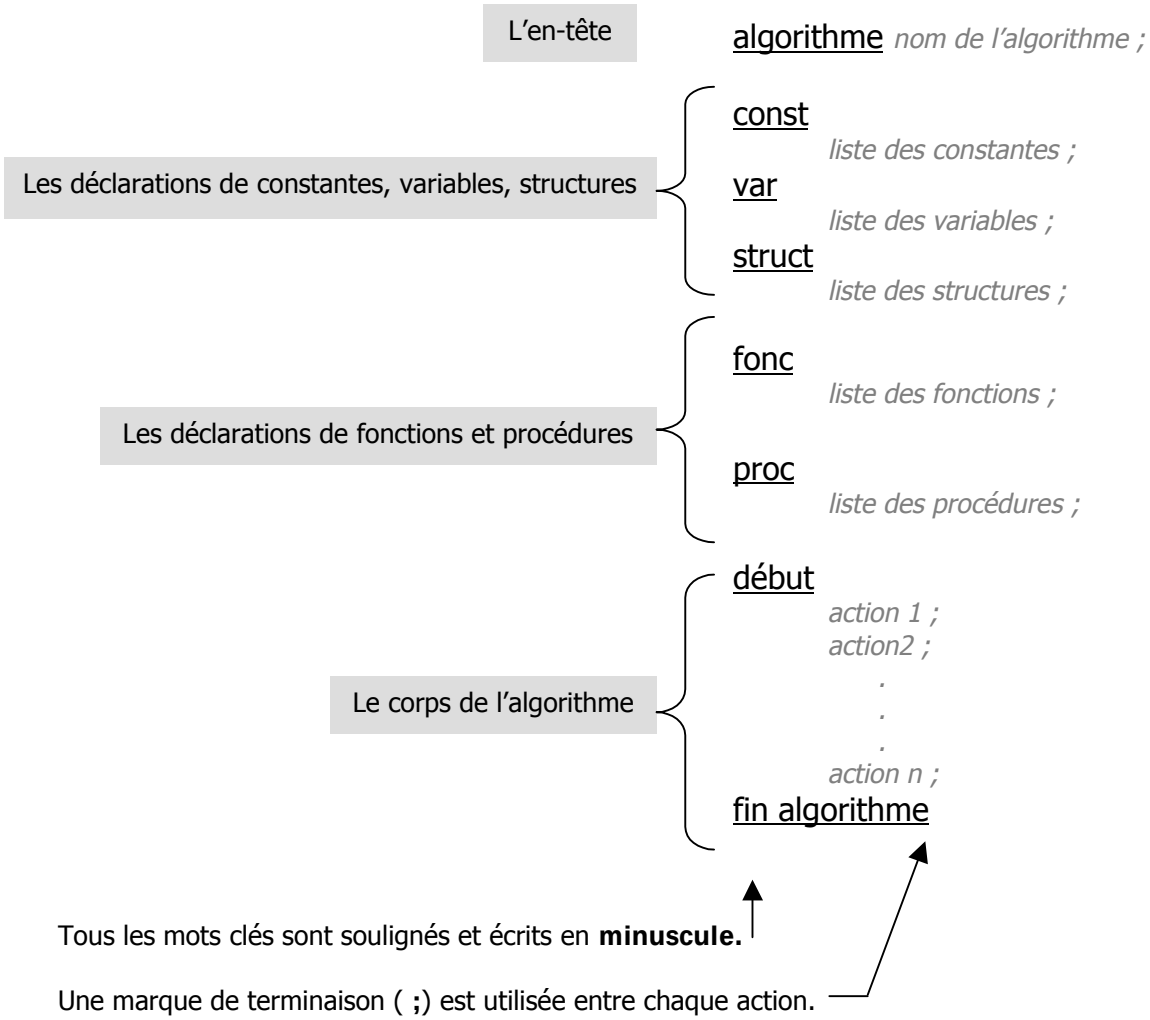
a. *Le langage de description d'algorithme*

Ce langage utilise un ensemble de *mots clés* et de *structures* permettant de décrire de manière complète, claire, l'ensemble des opérations à exécuter sur des données pour obtenir des résultats ; on n'hésitera donc pas à agrémenter l'algorithme de nombreux commentaires.

L'avantage d'un tel langage est de pouvoir être facilement transcrit dans un langage de programmation structuré (Pascal, C…)

III. Structure d'un algorithme

a. *Représentation :*



Tous les mots clés sont soulignés et écrits en **minuscule.**

Une marque de terminaison (**;**) est utilisée entre chaque action.

1. **L'en-tête**

   Il permet tout simplement d'identifier un algorithme.

2. **Les déclarations**

   C'est une liste exhaustive des objets, grandeurs utilisés et manipulés dans le corps de l ' algorithme ; cette liste est placée en début d'algorithme.

3. **Le corps**

   Dans cette partie de l'algorithme, sont placées les tâches (instructions, opérations…) à exécuter.

4. **Les commentaires :**

Pour permettre une interprétation aisée de l'algorithme, l'utilisation de commentaires est vivement conseillée.

Mot clé : co … fco

Voir exemple 2

IV. Déclaration de constantes, de variables et de structures :

a. Les constantes :

Elles représentent des chiffres, des nombres, des caractères, des chaînes de caractères, … dont *la valeur ne peut pas être modifiée* au cours de l'exécution de l'algorithme.

Mot clé : const

b. Les variables :

Elles peuvent stocker des chiffres des nombres, des caractères, des chaînes de caractères,… dont *la valeur peut être modifiée* au cours de l'exécution de l'algorithme.

Mot clé : var

Les constantes et les variables sont définies dans la partie déclarative par *deux caractéristiques essentielles*, à savoir :

- **L' identificateur** : c'est le nom de la variable ou de la constante. Il est composé de lettres et de chiffres
- **Le type** : il détermine la nature de la variable ou de la constante (entier, réel, booléen, chaîne de caractères…)

Voir exemple 1

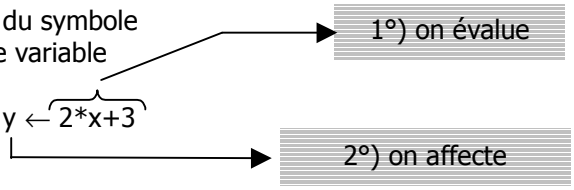
Pour pouvoir envisager des exemples d'utilisation, il faut introduire dès à présent l'instruction d'affectation ; elle s'écrit de la façon suivante :

identificateur de variable ← *valeur* ;

← : symbole d'affectation (ou assignation)

L'affectation se fait toujours en deux temps :

1. évaluation de l'expression située à droite du symbole
2. affectation du résultat à l'identificateur de variable



ainsi dans l'instruction d'affectation suivante $y \leftarrow 2*x+3$

c. Les structures :

Elles permettent de rassembler plusieurs variables ou constantes sous un même identificateur ; on parle aussi d'entités ou d'objets.

Mot clé : struct

V. Déclaration de procédures et de fonctions :

a. La procédure :

C'est un ensemble d'instructions référencé par un nom, et dont l' «exécution» est provoquée par le simple énoncé de ce nom.

Mot clé : proc

b. La fonction :

Comme pour la procédure, l' «exécution» d'une fonction est provoquée par la simple évocation de son nom, à la différence qu'elle se voit assigner une valeur dont le type doit être défini.

Mot clé : fonc

~

Procédures et fonctions sont des groupes de tâches à effectuer. L' intérêt de grouper ces tâches est de permettre :

- une lecture plus facile de l'algorithme principal(appelé également **ordonnancement**).
- de développer de manière indépendante des parties d'algorithmes dont l'emploi multiple au sein de l'algorithme principal est ainsi rendu plus aisé.

Voir exemple 2

1. Les paramètres :

Pour fournir à une procédure les informations qui doivent être traitées, et pour que la procédure puisse fournir en contrepartie des résultats, on utilise des paramètres. On distinguera trois types de paramètres:

- les *paramètres entrants* peuvent être consultés (et éventuellement modifiés) à l'intérieur de la procédure

- les *paramètres sortants* dont la valeur est déterminée à l'intérieur de la procédure et utilisable après l'appel à la procédure
- les *paramètres mixtes* ont une valeur à l'entrée dans la procédure, valeur qui peut être modifiée à l'intérieur de celle-ci, la modification étant répercutée à l'extérieur de la procédure

Voir exemple 2

VI. Les types de base

Nous avons vu qu'une des deux caractéristiques des constantes et des variables était leur type.

Nous considérerons cinq types de base :

1. L'entier

notation

45, 36, –564, 0 … en décimal
45h, 0FBh, 64h … en hexadécimal
% 10101111, %1011 … en binaire

Mot clé : entier

2. Le réel

-3.67, 4.2569, –564.0,18.36 10 e^{-6}…

Mot clé : réel

3. Le booléen

Il ne peut prendre que deux états : VRAI ou FAUX

Mot clé : booléen

4. Le caractère

'a', 'A','*','7','z',' !'….

Mot clé : car

5. La chaîne de caractères

'électronique', 'cd ROM de 80mn'…

Mot clé : chaîne

VII. Les opérateurs

a. Opérateurs sur les entiers et les réels

Arithmétiques	
+	Addition
-	Soustraction
*	Multiplication
/	Division
DIV	Division entière
↑	Puissance

comparaisons	
>	Supérieur
<	Inférieur
≥	Supérieur ou égal
≤	Inférieur ou égal
=	Egal
≠	Différent

b. Opérateurs sur les entiers et les booléens

Fonctions logiques	
Mot clé	
et	Fonction ET
ou	Fonction OU
oux	Fonction OU exclusif
non	Fonction NON
non et	Fonction NON ET
non ou	Fonction NON OU
>>	Décalage à droite
<<	Décalage à gauche

Fonctions de comparaison pour les booléens	
=	Egal
≠	Différent

c. Opérateurs sur les caractères et les chaînes

Fonctions de concaténation	
+	Concaténation

→ Voir exemple 3

Fonctions de comparaison pour les chaînes	
=	Egalité
≠	Différent
>	Supérieur
<	Inférieur

d. Priorité des opérateurs

Priorité à la multiplication et à la division.

e. L'affectation

Elle permet d'affecter une valeur à une variable.

Syntaxe : *identificateur de la variable* ← **expression** ;

L'expression est une suite d'opérations sur des constantes ou des variables déjà déclarées.

VIII. Les structures algorithmiques fondamentales :

Les opérations élémentaires relatives à la résolution d'un problème peuvent, en fonction de leur enchaînement être organisées suivant quatre familles de structures algorithmiques fondamentales.

- structures linéaires
- structures alternatives
- structures de choix
- structures itératives (ou répétitives)

a. Caractéristique de la structure linéaire

La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre énoncé.

Notation : faire action ;

Voir exemple 4

b. Caractéristique de la structure alternative

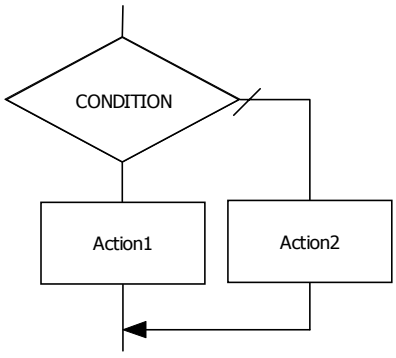
La structure alternative n'offre que deux issues possibles à la poursuite de l'algorithme et s'excluant mutuellement.

On peut rencontrer deux types de structures alternatives :

1. une structure alternative complète

Dans cette structure l'exécution d'un des deux traitements distincts ne dépend que du résultat d'un test effectué sur la condition qui peut être une variable ou un événement ;

- si la condition est vérifiée seul le premier traitement est exécuté ;
- si la condition n'est pas vérifiée seul est effectué le second traitement.

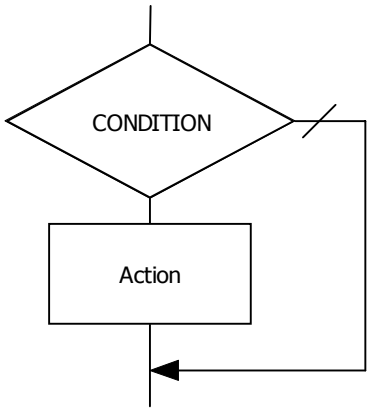


Notation :
si condition alors
action1 ;
sinon
action2 ;
fsi ;

Voir exemple 5

2. une structure alternative réduite

La structure alternative réduite se distingue de la précédente par le fait que seule la situation correspondant à la validation de la condition entraîne l'exécution du traitement, l'autre situation conduisant systématiquement à la sortie de la structure.



Notation : si condition alors
action ;
fsi ;

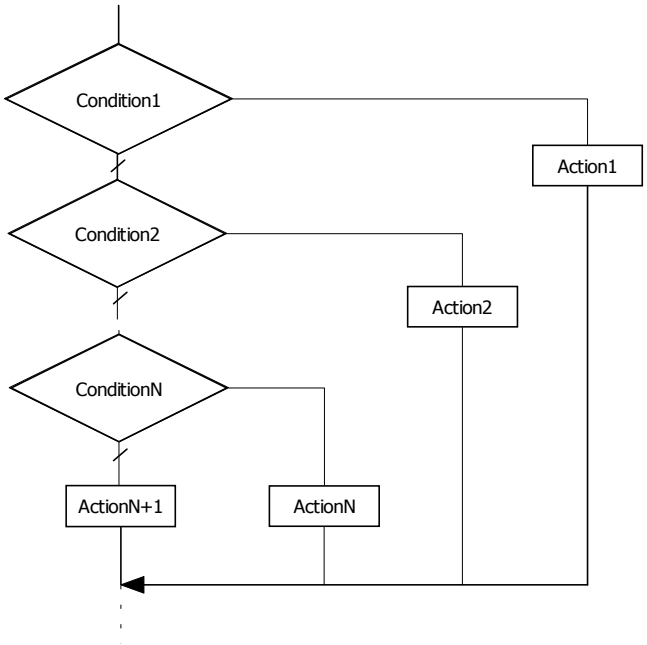
Voir exemple 6

c. Caractéristique de la structure de choix

La structure de choix permet, en fonction de plusieurs conditions de type booléen, d'effectuer des actions différentes suivant les valeurs que peut prendre une même variable.

Notation : suivant valeur faire
valeur1 : action1 ;
valeur2 : action2 ;
:
:
valeurN : actionN ;
sinon actionN+1 ;
fsuivant ;

Voir exemple 7



d. Caractéristique de la structure itérative

La structure itérative répète l'exécution d'une opération ou d'un traitement.

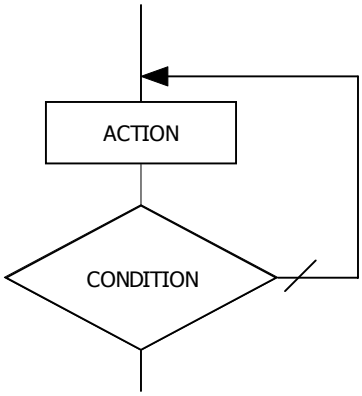
On considérera 2 cas :

premier cas : Le nombre de répétitions n'est pas connu ou est variable.

On distingue 2 structures de base :

structure RÉPÉTER JUSQU'À

Dans cette structure, le traitement est exécuté une première fois puis sa répétition se poursuit jusqu'à ce que la condition soit vérifiée.



Par traitement on entend :
- soit une structure isolée,
- soit une succession d'instructions.

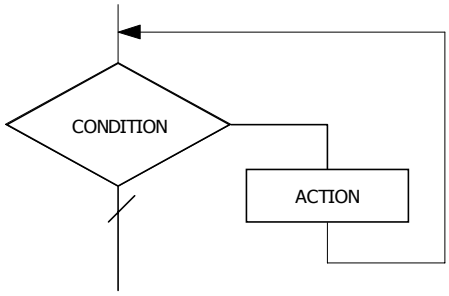
L'action est toujours exécutée au moins une fois.

Notation : répéter
action;
jusqu'à condition vraie ;

Voir exemple 8

structure TANT QUE … FAIRE…

Dans cette structure, on commence par tester la condition ; si elle est vérifiée, le traitement est exécuté.



Par traitement on entend :
- soit une structure isolée,
- soit une succession d'instructions.

L'action peut ne jamais être exécutée .

Notation : tant que condition faire
action;
ftant que ;

Voir exemple 9

deuxième cas : le nombre de répétitions est connu.

structure POUR…DE..À.. FAIRE…

Dans cette structure, la sortie de la boucle d'itération s'effectue lorsque le nombre souhaité de répétition est atteint.

On utilise donc une variable (ou indice) de contrôle d'itération caractérisée par :

- sa valeur initiale,
- sa valeur finale,
- son pas de variation.

*Si la valeur finale de l'indice est **inférieure** à sa valeur initiale le pas de variation est **négatif**, la structure est dite **« pour décroissante »*** (Figure 1)***;*** *dans le **cas contraire**, le pas est **positif** et la structure est dite **« pour croissante »*** (Figure 2)*.*

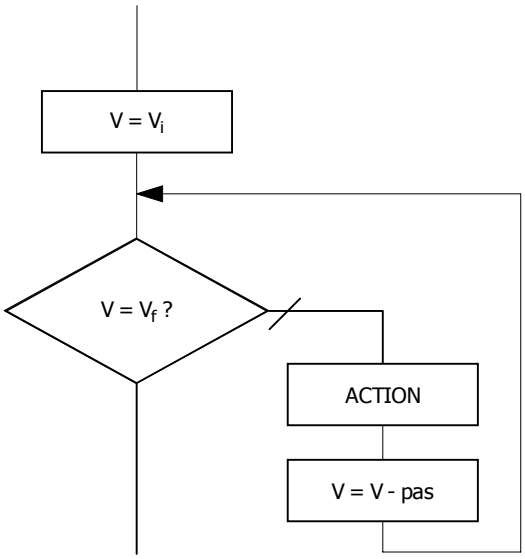


Figure 1

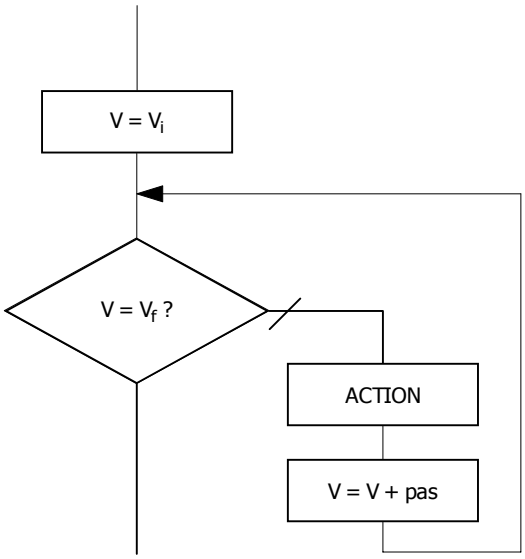


Figure 2

V : variable
Vi : valeur initiale de V
Vf : valeur finale de V

Notation : pour variable de début à fin pas n faire
action ;
fpour ;

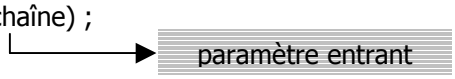
Voir exemple 10

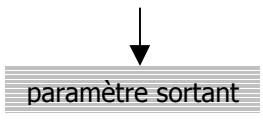
Exemple 1 :

```
algorithme exemple1 ;
const
        abscisse c'est 10 ;
        ordonnée c'est 30 ;
        vrai c'est 1 ;
        faux c'est 0 ;
var
        entier x, y;
struct
        disque  c'est
                entier abs,ord ; co centre du disque fco
                entier rayon ;
                entier couleur ;
                booléen visible ;
fstruct

début
…
fin algorithme exemple1.
```

Exemple 2 :

```
algorithme exemple2 ;
const
        …
var
        chaîne chaîne_lue ;
struct
        …
fstruct

proc
        afficher une chaîne de caractères(chaîne machaîne) ;
        …
fonc
        lire n caractères d'une chaîne(chaîne machaîne ,entier depuis la position , entier n) :chaîne ;
        …
début
        …
        chaîne_lue ← lire n caractères d'une chaîne(BEPEL ,0, 3) ;
        …
        afficher une chaîne de caractère(chaîne_lue) ;

fin algorithme exemple2.
```





Exemple 3 :

'A'+'ller' donne comme résultat après concaténation : 'Aller'

'alpha'+'numérique' donne comme résultat après concaténation : 'alphanumérique'

Exemple 4

Mise en marche d'un équipement

Avant de procéder à la mise en service d'un équipement, il est nécessaire d'effectuer un certains nombre d'opérations indispensables à son bon fonctionnement :

a) montée en température : *mise en route de l'accélérateur de chauffage AC ;*
b) distribution d'air comprimé : *ouverture de l'électrovanne d'admission EV.*
c) mise en route du dispositif de lubrification : *pompe d'arrosage P sous tension.*

Compléter l'algorithme suivant :

algorithme Mise en service d'un équipement ;

const Marche c'est 1 ;
Arrêt c'est 0 ;
var
AC co Accélérateur de chauffage fco ;
EV co Electrovanne d'admission d'air fco
P co Pompe de lubrification fco
EQ co Equipement fco

début

fin algorithme Mise en service d'un équipement .

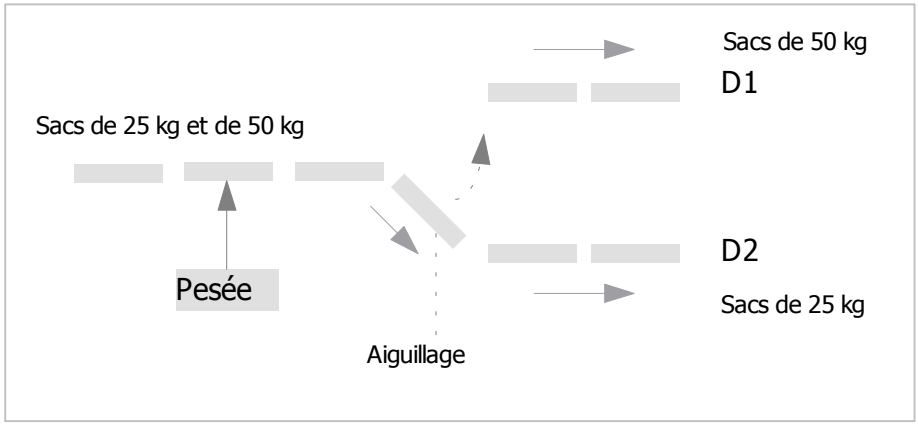
Construire l'algorigramme correspondant.

Exemple 5

Tri de sacs

A la sortie de l'atelier de conditionnement d'une usine de fabrication d'engrais, un même convoyeur à bande transporte indifféremment des sacs de 25 kg et des sacs de 50 kg.

Un dispositif de tri automatique dirige ces sacs vers deux zones distinctes de stockage **D1** et **D2**.



Construire l'algorithme correspondant :

algorithme tri automatique;

const

var

début

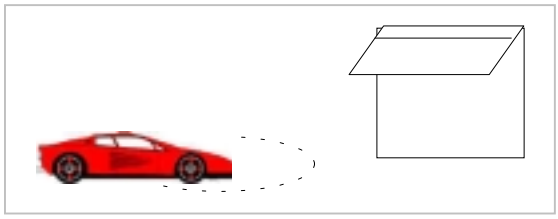
fin algorithme tri automatique.

Construire l'algorigramme correspondant.

Exemple 6

Ouverture de la porte d'un garage

Le capteur de présence de la voiture du propriétaire du garage déclenche l'ouverture automatique de la porte et uniquement dans ce cas.



Construire l'algorithme correspondant :

algorithme ouverture automatique;

const

var

début

fin algorithme ouverture automatique.

Construire l'algorigramme correspondant.

Exemple 7

Exemple 8

Utilisation d'un four à micro ondes

Un four à micro ondes doit fonctionner pendant un temps t_f égal au temps t_p, programmé par l'utilisateur.

Compléter l'algorithme suivant :

algorithme durée de fonctionnement du four ;

var
t_f co temps de chauffe fco ;
t_p co temps programmé fco
fonc
mesurer le temps de chauffe() : tf ;

début

fin algorithme durée de fonctionnement du four.

Construire l'algorigramme correspondant.

Exemple 9

Embouteillage

Dans une usine de fabrication de jus de fruits, les bouteilles sont conditionnées par six, après contrôle, sous un film plastique rétractable:

Compléter l'algorithme suivant :

algorithme Mise sous film plastique ;

var
V co Nombre de bouteilles à conditionner fco

proc
contrôler() ;
début

fin algorithme Mise sous film plastique .

Construire l'algorigramme correspondant.