REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE FERHAT ABBAS – SETIF 1 –

Faculté de Technologie

Département d'Enseignement de Base en Technologie

Cours informatique 1

Dr. Sadik BESSOU

Maitre de conférences en Informatique

1er année tronc commun de Technologie

2015/2016

بسم الله الرحمن الرحيم

والصلاة والسلام على رسول الله 

Table des matières

Introduction

Ce manuel expose l'algorithmique non seulement d'un point de vue théorique, mais aussi sous l'angle pratique, en tant qu'outil pour la modélisation et la résolution des problèmes. Il propose en outre de nombreux exercices avec solutions et des exemples illustratifs, des problèmes ouverts et pistes de réflexions originaux et pédagogiques en fin de chaque chapitre.

Objectifs du cours

L'apprenant sera capable :

- d'analyser tous types de problèmes en passant par les trois étapes d'analyse
- de décrire les étapes de résolution d'un problème
- de formuler les étapes de n'importe quelle solution sous forme algorithmique
- de maitriser les techniques de programmation
- de réécrire un algorithme en un langage de programmation

Pré requis

Pour pouvoir suivre le cours il faut :

- Savoir les bases de mathématiques
- Avoir un raisonnement logique

Contenu de cours

Ce cours est destiné à vous faire acquérir les bases de l'algorithmiques, c'est-à-dire à vous apprendre comment écrire de façon claire et concise un algorithme qui décrit les étapes de résolution d'un problème donné.

Il s'agit d'analyser le problème, distinguer les différentes étapes de résolution et décomposer le problème en sous problèmes puis les formuler en une structure algorithmique.

Le manuel est présenté en 5 chapitres :

Chapitre 1 : Notions algorithmiques

Chapitre 2 : Structure d'un algorithme

Chapitre 3 : Les structures de contrôle conditionnelles

Chapitre 4 : Les structures de contrôle répétitives

Chapitre 5 : Mini Projet

Langage de programmation

Le langage de programmation adopté pour notre cours est Pascal, vu qu'il est pédagogique et très proche de l'algorithmique. Apres avoir maitrisé pascal, l'étudiant pourra apprendre n'importe quel autre langage facilement.

Évaluation

Evaluation continue par des travaux pratiques et des mini-projets.

Evaluation sommative en fin de semestre.

Chapitre 1. Notions algorithmiques

1. Introduction

L'objectif de ce chapitre est de comprendre les notions de base d'algorithmique, et la maitrise de la terminologie, ainsi de se rappeler de quelques notions mathématiques nécessaires pour l'écriture des algorithmes.

2. Histoire et étymologie

Le mot algorithme vient du nom du mathématicien *Mohamed Ibn Moussa Al Khawarizmi* qui a proposé une méthode d'enseignement de calcul aux enfants, cette méthode est caractérisée par l'enchainement et le détail des étapes, cette même idée est réutilisée pour enseigner l'ordinateur à résoudre des problèmes (vue les similitudes entre l'enfant ou l'apprenant et l'ordinateur).

3. Définition

Un algorithme est un ensemble d'étapes détaillées pour la résolution d'un problème donnée.

4. Les objets de base

Les objets de base qu'on utilise dans les algorithmes sont les suivants :

4.1.Les constantes

Sont des objets caractérisés par :

1. Un nom fixe
2. Un type fixe
3. Une valeur **fixe**

Une valeur est donnée à cet objet au début de l'algorithme.

Exemple

La gravitation **G** = 10, la valeur de π = 3.14,…

4.2.Les variables

Sont des objets caractérisés par :

1. Un nom fixe
2. Un type fixe
3. Une valeur **variable**

Les variables sont déclarées au début de l'algorithme (on ne peut pas utiliser une variable avant de la déclarer)[1]

Les valeurs des variables sont entrées par l'utilisateur.

Exemple

P : est une variable représentant le poids, T : est une variable représentant la taille,…

4.3.Les expressions

Les expressions sont des objets constitués de constantes et/ou de variables liées par des opérateurs arithmétiques ou logiques.

On peut les décomposer en deux types : les expressions arithmétiques et les expressions logiques.

Les expressions arithmétiques

Elles comportent des opérateurs arithmétiques (+, -, *, /, div, mod)

Exemple

A + b - c, a * b – c * d

L'évaluation de ce type d'expressions renvoie une valeur numérique.

Les expressions logiques

Elles comportent des opérateurs logiques, elles sont décomposées en deux :

Les expressions logiques simples

Elles comportent les opérateurs suivants : ($>, \geq, <, \leq, =, \neq$)

Exemple

$a > b$, $7 \geq 5$,$10 = 8$,…

Leur évaluation donne une valeur logique : vrai ou faux.

Les expressions logiques composées

Elles sont constituées d'expressions simples ou des variables logiques liées par des opérateurs logiques (et, ou, non)

Exemple

[1] Dans certains langages de programmation il est possible.

(a > b) et (c = d), A et non (B et C ou A)

Leur évaluation donne une valeur logique : vrai ou faux.

Table de vérité

Pour évaluer les expressions logiques, on utilise la table de vérité suivante :

A	B	A et B	A ou B	Non A
Faux	Faux	Faux	Faux	Vrai
Faux	Vrai	Faux	Vrai	Vrai
Vrai	Faux	Faux	Vrai	Faux
Vrai	Vrai	Vrai	Vrai	Faux

Question

Quel est le résultat de l'expression suivante :

E= 7+5*2

Est-ce que : E = 24 ? ou E =17 ?

Si on commence par l'addition, puis la multiplication, on obtient : 24

Si on commence par la multiplication, puis l'addition, on obtient : 17

Mais pour toute expression, il faut trouver une solution unique. Si on donne cette expression à l'ordinateur, il nous donne : 17, pour quoi?

Réponse

Le résultat est 17 car l'ordinateur a un ordre d'évaluation des expressions; il a une priorité des opérateurs.

Priorités des opérateurs

On évalue les expressions avec l'ordre suivant, (ordre descendant de priorité) :

1- Parenthèses
2- Non, le moins unitaire
3- *, /, div, mod, et
4- +, -, ou
5- >, ≥, <, ≤, =, ≠.

En cas d'égalité, on commence par l'opération la plus à gauche (si on écrit en français et par l'opération la plus à droite si on écrit en arabe).

Remarque

Les parenthèses peuvent être utilisées pour modifier les priorités d'évaluation.

Exemple

E= 7+5*2

Pour pousser le programme à commencer par l'addition avant la multiplication, on rajoute des parenthèses.

E= (7+5)*2

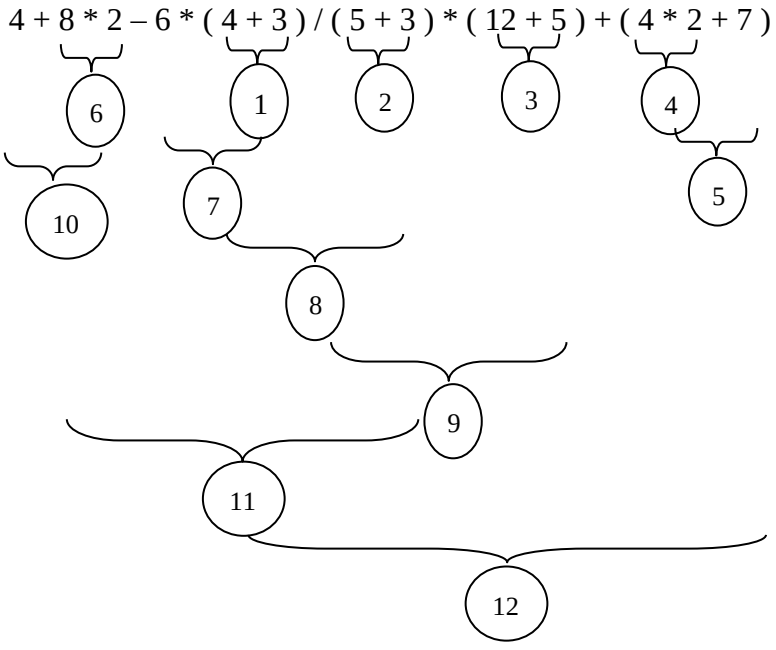
Exercice

Evaluer les expressions suivantes :

1. E= 4+8*2-6*(4+3)/(5+3)*(12+5)+(4*2+7)
2. E= 14 mod 9 div 2 + 14 mod 5 mod 3
3. E= (12<3) OU (14+2*3>16-8/4)
4. E= FAUX OU NON VRAI ET NON(FAUX OU VRAI)
5. E= FAUX ET (FAUX ET VRAI OU NON FAUX ET NON(NON(VRAI ET FAUX)OU NON(VRAI ET NON VRAI)))

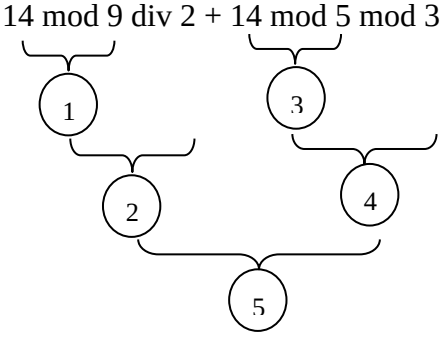
Solution

1.



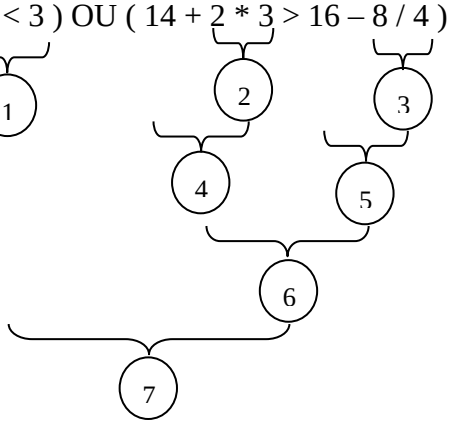
E= 124.25

2.



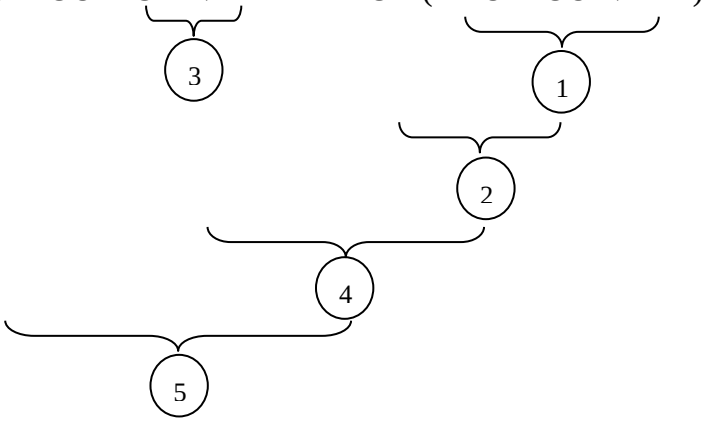
E= 3

3. (12 < 3) OU (14 + 2 * 3 > 16 – 8 / 4)



E= Vrai

4. FAUX OU NON VRAI ET NON (FAUX OU VRAI)



E= Faux

5. FAUX ET (FAUX ET VRAI OU NON FAUX ET NON(NON(VRAI ET FAUX)OU NON(VRAI ET NON VRAI)))

E=FAUX

Pour cette expression, on peut ne pas l'évaluer car nous avons FAUX et une deuxième partie entre parenthèse, que ce soit la valeur de la deuxième partie le résultat sera FAUX : FAUX ET VRAI → FAUX, FAUX ET FAUX → FAUX.

Règles de choix des noms des objets

Il y a cinq règles pour bien nommer un objet (constante, variable ou expression) :

1- Le nom doit commencer par une lettre alphabétique suivie de lettre ou de chiffres (A, b, b23, poids, nom_etud, et non 1b, 3c,..).

2- Le nom doit ne pas contenir des caractères spéciaux (?, !, <, >, §, #, -,…), le caractère (_) n'est pas considéré comme caractère spécial donc le nom (A_5) est valable.

3- Le nom doit être diffèrent des mots réservés par le langage de programmation (le nom *begin* est interdit en *Pascal*, le nom *print* est interdit en *Python*,…).

4- Le nom doit être diffèrent des autres noms dans le même algorithme (on ne peut pas donner le même nom à deux variables)

5- Il est souhaitable que le nom y un sens pour le distinguer des autres variables (pour la variable qui représente le poids on peut proposer poids, p,… mieux que : a, b, …)

4. Les types de base

Chaque objet à un type invariable au cours de l'algorithme. Les types en algorithmiques sont différents de ceux des mathématiques (naturels, entiers, réels,…).

4.1.Entier

Est le type affecté aux objets qui représentent une quantité indivisible (leurs valeurs possibles ne comportent pas de virgule).

Exemple

Age, année, nombre d'étudiants,…

Les valeurs possibles pour un nombre d'étudiants sont par exemple : 30 étudiants, 40 étudiants, 1000 étudiants et non pas 7.5 étudiants.

Remarque

De façon générale, les entiers sont compris entre -32768 et 32767.

En pascal

Le type entier est appelé **Integer**.

4.2.Réel

Est le type affecté aux objets fractionnaires (leurs valeurs possibles comportent une virgule)

Exemple

Note d'un étudiant, longueur d'un rectangle,…

Les valeurs possibles pour une note d'un étudiant sont par exemple : 10, 12.5, 8.75,…

En pascal

Le type réel est appelé **real**.

4.3.Caractère

Est le type affecté aux objets qui ont pour valeur une lettre alphabétique (A…Z, a…z) ou un caractère spécial (?, +, -, *, /, #,…) ou un chiffre (1…9)

Exemple

Le nom d'un groupe

Les valeurs possibles pour un groupe sont par exemple : A, B, H,…

En pascal

Le type caractère est appelé **char**.

4.4.Chaine de caractères

Est le type affecté aux objets qui ont pour valeur un ensemble de lettres ou de caractères.

Exemple

Un nom d'étudiant, nom d'une ville,…

Les valeurs possibles pour un nom d'étudiant sont par exemple : Salim, Walid, Nabil, …

En pascal

Le type chaine de caractères est appelé **string**.

4.5.Booléen

Est appelé aussi le type logique. Est le type affecté aux objets qui ont pour valeur vrai ou faux

Exemple

Marié, admis, trouvé

Les valeurs possibles pour admis sont : vrai, faux

En pascal

Le type logique est appelé **boolean**.

5. Les instructions de base

Ce sont les instructions qui manipulent les objets de base.

5.1.L'instruction de lecture (lire)

Elle permet à l'utilisateur de rentrer une valeur par clavier, cette valeur sera affectée à une variable.

Exemple

Lire (x), lire (poids), lire (note)

Il est possible de lire plusieurs variables soit dans la même instruction ou dans plusieurs instructions

Exemple

Lire (a, b, c) ou lire (a)
lire (b)
lire (c)

On ne lit que des variables et non pas des constantes ou des expressions.

Lire (10) ou lire (a+b) sont impossibles.

En pascal

L'instruction lire est représentée par *read* ou *readln*. L'instruction *read* lit une valeur, et le curseur reste dans la même ligne.

Readln agit exactement comme *read*, mais ajoute un retour à la ligne après la lecture.

Exemple

```
Readln(X);
Read(Y);
```

5.2.L'instruction d'affectation (←)

Est l'instruction qui permet de stocker une valeur dans une variable.

Exemple

A← 5 (A reçoit 5)
S← A
b← 5 + 10
c← 'A'

L'objet à gauche de la flèche doit être une variable, par contre l'objet à droite de la flèche peut être une variable, une constante ou une expression.

Alors a + b← 7 ou 5← b sont incorrectes.

Le type de la variable à gauche de la flèche et le type de la variable ou la constante ou l'expression à droite de la flèche doivent être du même type.

Remarque on peut affecter une valeur entière à une variable réelle, ainsi, on peut affecter un caractère à une variable de type chaine de caractères. Car l'ensemble des réels contient l'ensemble des entiers, ainsi l'ensemble des chaines de caractères contient l'ensemble des caractères.

En pascal

L'instruction ← est représentée par **:=**

Exemple

```
A := 8;
```

C := 2.3;

5.3.L'instruction d'affichage (écrire)

Elle permet à l'utilisateur de visualiser la valeur d'une variable ou d'une constante ou d'une expression sur écran.

Exemple

Ecrire (x), Ecrire (Pi), Ecrire (a+b)

On peut afficher un message expressif avec une variable :

Ecrire ('La moyenne = ', moy)

Si la variable est de type réel le résultat sera affiché par exemple comme suit :

1.000000000000E10

Pour le convertir en un affichage lisible, on rajoute deux chiffres après la variable

Ecrire (moy:5:2), 5 veut dire : on réserve 5 positions à l'écran pour écrire la variable, et le 2 veut dire : on affiche 2 chiffres après la virgule.

En pascal

L'instruction lire est représentée par *write* ou *writeln*. L'instruction *write* lit une valeur, et le curseur reste dans la même ligne.
writeln agit exactement comme *write*, mais ajoute un retour à la ligne après l'affichage.

Exemple

Write (A);
Writeln (B);
On peut afficher un message expressif avec la variable en le délimitant entre apostrophes :

Writeln ('La somme = ', S);
Writeln ('Ce nombre apparait ', n, ' fois');

6. Conclusion

Nous avons vu dans ce chapitre les objets de base qu'on manipule dans la programmation qui sont les variables, les constantes et les expressions. Les types de bases pour ces objets sont : entier, réel, caractère, chaine de caractères, booléen. Les instructions de base sont : l'instruction de lecture, d'affectation et d'affichage. A la fin de chaque section on a donné la traduction en Pascal des différents mots réservés.

7. Série d'exercices N°=1

Exercice 1.1

Indiquer les erreurs si elles existent dans les identificateurs des objets suivants :

Note-math	année	2016_cours	nom d'étudiant	faux	var
Erreur	n6	Not	12.85	β	MoySem

Exercice 1.2

Evaluer les expressions arithmétiques suivantes, tel que A=8, B=12 :

1- 5-4*3+12*5-(2+5)*(9+3-2*5)
2- -15+2*14/7+(5+12-9)*(8+33.3)/4
3- 24 mod 2+12 * 16 div 5*2/3+11 div (5 div 2)
4- A+B/B-2*A+(B-A)
5- B/3+15 div 4 mod 2 –(7*(28-B)+(A*4mod B))

Exercice 1.3

Evaluer les expressions logiques suivantes, tel que A=Vrai, B=Faux :

1- A OU B
2- Non A OU B ET A
3- (A OU B) ET (A OU NON A)
4- NON ((NON A ET B) OU (A ET NON B))
5- (NON (A ET B) ET A) OU (NON (A ET B) ET B)

Exercice 1.4

Donner la déclaration des objets suivants, sachant que A et B sont des entiers.

C←A / B
D←25.5
H←'H'
K←C + A - B
M←A + 3 * B
J←A mod B
M←C + 2
H←'Admis'

F←D
S←J > 10

Exercice 1.5

Donner la déclaration appropriée des objets suivants :

Vitesse, lieu de naissance, âge, moyenne, longueur, nom d'une personne, groupe sanguin, nombre d'étudiant, variance, valeur absolue, distance, observation (admis ou non).

Exercice 1.6

Donner la valeur et le type de chacune des expressions suivantes sachant que : A= 13 et B=4

A+B	A-B	A*B	A/B	A div B	A mod B
Sin(B)	A/3	A+3*B>12	Abs (B-A)	A-B=0	3*A+B/5

Exercice 1.7

Donner les valeurs des variables après l'exécution des instructions suivantes :

C←12
B←C +14
D←B / A
E←B * 3
B← C mod A

Exercice 1.8

Déterminer l'erreur si elle existe dans chaque instruction (les instructions sont indépendantes)

A←A
A←'A'
'A'←A
A←A + A
A+B←A
A←A + B
B←B > A
B←C < A

Chapitre 2. Structure d'un algorithme

1. Introduction

L'objectif de ce chapitre est d'exploiter les notions vues en chapitre 1 pour formuler la solution en un formalisme conventionnel qui est l'algorithme. La trace d'exécution est expliquée pour bien appréhender le déroulement d'un algorithme et assurer son exactitude. La traduction en Pascal est donnée chaque fois il y un nouveau terme ou concept.

2. Analyse de problème

Avant de résoudre n'importe quel problème, il faut passer d'abord par une analyse. Cette analyse permet d'indiquer les objets manipulés, leurs types, et les instructions correspondantes dans l'algorithme.

Etapes d'analyse

1- Citer les entrées

2- Citer les sorties

3- Citer les opérations

Classer les objets en variables et en constantes et indiquer leurs types.

Exemple

Analyser les problèmes suivants :

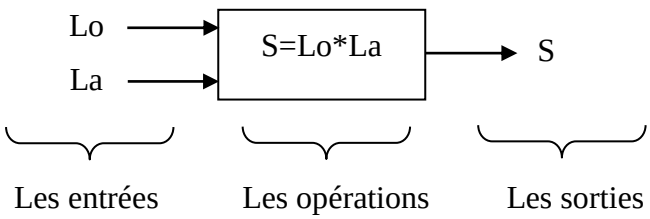
1- Calculer la surface d'un rectangle ?

2- Calculer l'aire d'un cercle ?

3- Calculer le volume d'un cylindre ($V=pi * r^2 * h$)

4- Calculer l'énergie cinétique d'un corps en mouvement ? ($E_c = 1/2 * m * v^2$)

2.1.La surface d'un rectangle

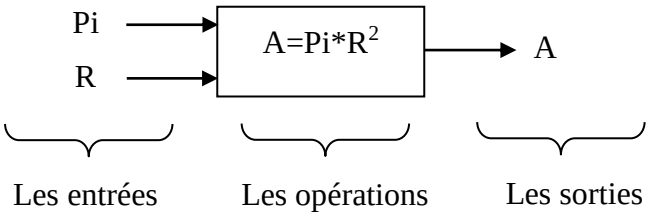


Tous les objets sont des variables. Lo, La, S : réel.

Remarque

Le nombre des entrées peut être 0 ou plus, le nombre des sorties peut être 0 ou plus.

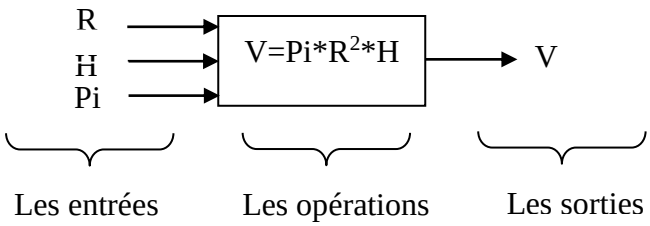
2.2.L'aire d'un cercle



Les variables sont : R et A de type réel

Les constantes sont : Pi=3.14

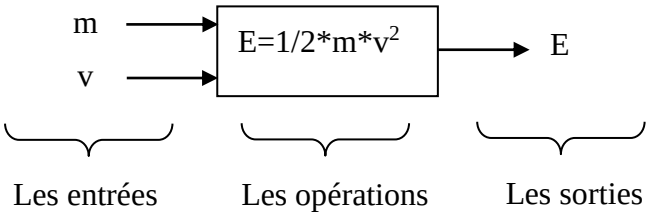
2.3.Le volume d'un cylindre



Les variables sont : R, H et V de type réel

Les constantes sont : Pi=3.14

2.4.L'énergie cinétique



Les variables sont : m, v et E de type réel

Les constantes sont : Pi=3.14

3. Forme d'un algorithme

Algorithme nom_algorithme
Constantes nom_const = valeur
Variables nom_var : type
Début
……
Liste d'instructions
…….
Fin

Une bonne analyse conduit à un algorithme correct

Exploitation de l'analyse

Apres avoir effectué l'analyse de problème, l'écriture de l'algorithme s'accomplit de façon systématique comme suit :

1- Toutes les entrées et les sorties sont déclarés au début de l'algorithme
2- Toutes les entrées sont lues par (lire)
3- Toutes les opérations sont réalisées par (←)
4- Toutes les sorties sont affichés par (écrire)

Exercice 1

Prenons l'analyse du problème 3 pour écrire l'algorithme correspondant :

Solution

Algorithme cylindre
Constantes Pi=3.14
Variables R, H : réel
Début
Lire (R)
Lire (H)
V← Pi * R * R * h
Ecrire (V)
Fin

Exercice 2

Ecrire un algorithme qui lit une température en degré *Celsius* (°C) et la convertit en *Kelvin* (K) en utilisant la formule : $\mathbf{T_K = T_C + 273,15}$

Solution

Algorithme conversion
Constantes n=273.15
Variables T, K : réel

Début
    Lire (T)
    K← T + n
    Ecrire (K)
Fin

Exercice 3

Ecrire un algorithme qui calcule le volume d'une pyramide à base carrée :

$$V = (C^2 * h) / 3$$

Solution

Algorithme pyramide
Variables c, h,V : réel
Début
    Lire (c)
    Lire (h)
    V← (c * c * h) / 3
    Ecrire ('Le volume = ', V)
Fin

4. Traduction en Pascal

La forme générale d'un programme Pascal est comme suit :

Program nom;
Const nom_const=valeur;
Var nom_var : type;
Begin
    Inst;
End.

Chaque instruction doit se terminer par un point-virgule, sauf les exceptions (comme le mot *begin*, le mot *end* final)

Exercice 4

Traduire l'algorithme de l'exercice précédent en Pascal

Solution

Program pyramide;
Var c, h,V : real;
Begin
    readln (c);
    readln (h);
    V:= (c * c * h) / 3;
    writeln (V);
End.

5. Trace d'exécution

La trace d'exécution d'un algorithme est un tableau avec une colonne par variable, où l'on écrit l'évolution des variables pendant le déroulement de l'algorithme, et une colonne pour l'affichage sur écran. Les instructions *(lire*) et (←) sont inscrites sur la première partie (mémoire) et l'instruction *(écrire*) est inscrite sur la deuxième partie (écran).

Mémoire			Ecran
X	Y	Z	aaaaaaa
…	…	…	
…	…	…	
…	…	…	
…	…	…	

Exemple

Tracer le tableau d'exécution de l'algorithme suivant pour A= 18 et B =5

Algorithme trace
Var A, B, C : entier
X, Y : réel
Début
1. Lire (A)
2. Lire (B)
3. C←A + B
4. X←A / B
5. A←A - B
6. Y←X * 3
7. A←B + A mod B
8. X← Y - X
9. Ecrire (A, B, C, X, Y)

Fin

N°	Mémoire					Ecran				
	A	B	C	X	Y					
1	18	?	?	?	?					
2	18	5	?	?	?					
3	18	5	23	?	?					
4	18	5	23	3.6	?					
5	13	5	23	3.6	?					
6	13	5	23	3.6	10.8					
7	8	5	23	3.6	10.8					
8	8	5	23	7.2	10.8					
9	8	5	23	7.2	10.8	8	5	23	7.2	10.8

6. Conclusion

Nous avons étudié dans ce chapitre les étapes d'analyse d'un problème, puis la forme générale d'un algorithme, avec quelques exemples. En fin nous avons vu les étapes de trace d'exécution; opération essentielle pour valider l'algorithme.

7. Série d'exercices N°=2

Exercice 2.1

Ecrire un algorithme pour convertir une température exprimée en *Fahrenheit* en degré *Celsius*, en utilisant la formule suivante :

$$C° = (5/9)\,(F°-32)$$

Exercice 2.2

Ecrire un algorithme qui calcule le volume et l'aire d'une sphère en utilisant les expressions suivantes :

$$V = 4\pi r^3/3$$

$$A = 4\pi r^2$$

Exercice 2.3

Ecrire un algorithme qui calcule la masse d'air contenue dans un pneu d'automobile, en utilisant la formule suivante :

$$m = 350(PV)/(T+273)$$

Avec P : la pression (bar)

V : le volume (m^3)

M : la masse d'air (Kg)

T : la température (degré Celsius)

Exercice 2.4

Ecrire un algorithme qui calcule la consommation d'un véhicule pour 100 Km

Consommation (aux 100 km) = $\dfrac{\text{Nombre de litres du plein effectué x 100 km}}{\text{Nombre de km parcourus depuis le dernier plein}}$

Exercice 2.5

Ecrire un algorithme qui calcule le périmètre crânien (PC) d'un bébé: PC= (T/2) +10

Exercice 2.6

Ecrire un algorithme qui calcule l'apport énergétique d'un repas, sachant que :

Glucides : 4Kcal/g, Protides : 4Kcal/g, Lipides : 9Kcal/g

Exercice 2.7

Ecrire un algorithme qui calcule le taux de LDL, tel que :

LDL = Cholesterol total – HDL – TG/5

Exercice 2.8

Ecrire un algorithme qui calcule la pression artérielle moyenne :

P= (Pression systolique + 2 * Pression diastolique) / 3

P systolique : P max : contraction du cœur

P diastolique : P min : relâchement du cœur

Exercice 2.9

Ecrire un algorithme qui demande à l'utilisateur les valeurs de 2 entiers x, y et permute leurs valeurs et les affiche.

Exercice 2.10

Ecrire un algorithme qui demande à l'utilisateur les coordonnées de 2 points distincts du plan et qui affiche les coordonnées du point milieu.

Exercice 2.11

Ecrire un algorithme qui demande à l'utilisateur une valeur pour U_0, r et n et qui affiche la énième valeur de la suite arithmétique définie par :

U_0 et $U_{n+1} = U_n + r$.

(On rappelle la propriété : $U_n = U_0 + n * r$)

Exercice 2.12

Au temps $t_0 = 0$, un obus est lancé verticalement en l'air à partir d'une plate-forme située à Y_0 mètres du sol avec une vitesse initiale de V_0 mètres par seconde.

En prenant G= 9.81, écrire un algorithme qui demande à l'utilisateur les valeurs de Y_0, V_0, une valeur de temps t et qui affiche la vitesse V ainsi que la hauteur Y de l'obus par rapport au sol précisément à l'instant t.

Exercice 2.13

Ecrire un programme qui demande à l'utilisateur la valeur d'une durée exprimée en secondes et qui affiche sa correspondance en heures minutes secondes.

Exemple : 3800 s → 1 heure 3 minutes 20 secondes.

Exercice 2.14

On considère la suite d'instructions :

X← X + Y

Y← X - Y

X← X - Y

Que contiennent les variables numériques X et Y après cette série d'affectations, en supposant que X et Y valaient 5 et 2?

D'après vous, quel est le résultat obtenu pour des valeurs numériques quelconques ? Justifiez

Proposer une autre façon d'effectuer ce type d'opérations.

Exercice 2.15

Ecrire un algorithme qui permet d'effectuer une permutation circulaire des valeurs entières de 3 variables X, Y, Z (X ← Y, Y ← Z, Z ← X)

Exercice 2.16

Tracer le tableau d'exécution de l'algorithme suivant pour A= 12 et B=10

```
Algorithme exo12
Constantes A=10
Var B, C : entier
    D, E : réel
Début
    Lire (A, B)
    A←A * B
    B←A * B
    Ecrire (A, B)
    B←A mod B
    A← B mod A
    Ecrire (A - B, A / B)
    Ecrire (A, B)
Fin
```

Chapitre 3. Les structures de contrôle conditionnelles

1. Introduction

Le présent chapitre expose une structure de donnée primordiale qui est la structure alternative; cas omniprésent dans la majorité des problèmes, chaque problème a plusieurs possibilités de solutions selon les données. Les structures alternatives ou conditionnelles répondent à cette problématique.

Exercice 1

Ecrire un algorithme qui lit un nombre entier et calcule son carrée ?

Solution

Algorithme exo1
Var N, R : entier
Début
    Lire (N)
    R← carree (N)
    Ecrire (R)
Fin

Exercice 2

Ecrire un algorithme qui lit un nombre entier et calcule sa racine carrée ?

Solution

Algorithme exo2
Var N : entier, R : réel
Début
    Lire (N)
    R← racine (N)
    Ecrire (R)
Fin

Cette solution est fausse !, car il y a des cas qui ne marchent pas : le cas où N est négatif, l'algorithme ne marche pas. Donc, quelle est la solution ?

Réponse

Utiliser les structures conditionnelles.

2. Les structures conditionnelles simples

Règle

On utilise les structures conditionnelles quand il y a plusieurs possibilités dans la solution (deux ou plus)

Structure

Le test s'écrit comme suit :

```
Si (condition) alors
    Instructions 1
Sinon
    Instructions 2
Fsi
```

L'instruction 1 n'est exécutée que si la condition est vraie. Si la condition est fausse, l'instruction 2 est exécutée. On exécute une seule instruction selon la condition.

Donc la solution de l'exercice précèdent sera comme suit :

```
Algorithme exo2
Var N : entier, R : réel
Début
    Lire (N)
    Si N > 0 alors
        R← racine (N)
        Ecrire (R)
    Sinon
        Ecrire ('Erreur')
    Fsi
Fin
```

Exercice 3

Ecrire un algorithme qui lit un nombre entier et calcule son inverse ?

Solution

```
Algorithme exo3
Var N : entier, I : réel
Début
    Lire (N)
    Si N≠0 alors
        I← 1 / N
        Ecrire (I)
    Sinon
        Ecrire ('Erreur')
    Fsi
Fin
```

On peut exécuter une instruction si la condition est vraie, et on ne fait rien sinon.

```
Si condition alors
    Instructions
Fsi
```

Exercice 4

Ecrire un algorithme qui lit un nombre entier et affiche sa valeur absolue ?

Solution

```
Algorithme exo4
Var N : entier
Début
    Lire (N)
    Si N < 0 alors
        N← -1 * N
    Fsi
    Ecrire (N)
Fin
```

Exercice 5

Ecrire un algorithme qui lit un nombre entier et affiche si ce nombre est pair ou impair ?

Solution

```
Algorithme exo5
Var N : entier
Début
    Ecrire ('Taper un nombre entier')
    Lire (N)
    Si N mod 2 = 0 alors
        Ecrire (N,' est pair')
    Sinon
        Ecrire (N, ' est impair')
    Fsi
Fin
```

3. Les structures conditionnelles complexes

On utilise les structures conditionnelles complexes ou imbriquées si le nombre de cas est supérieur à 2. Donc, on préserve la règle : (si le nombre de cas restants est supérieur à 2 utiliser un Si).

Exercice 6

Ecrire un algorithme qui lit un nombre entier et affiche si ce nombre est positif, négatif ou nul

Solution

```
Algorithme exo6
Var N : entier
Début
    Ecrire ('Taper un nombre entier')
    Lire (N)
    Si N > 0 alors
        Ecrire (N, ' est positif')
    Sinon
        Si N < 0 alors
            Ecrire (N, ' est négatif')
        Sinon
            Ecrire (N, ' est nul')
        Fsi
    Fsi
Fin
```

Exercice 7

Ecrire un algorithme qui lit une moyenne d'un étudiant et affiche son observation ?

Moy dans : [0-5[→ Très faible

Moy dans : [5-8[→ Faible

Moy dans : [8-12[→ Bien

Moy dans : [12-15[→ Très bien

Moy dans : [15-20] → Excellent

Solution

```
Algorithme exo7
Var Moy : réel
Début
        Ecrire ('Taper une moyenne')
        Lire (Moy)
        Si Moy < 0 alors
                Ecrire ('Erreur')  puisque
        Sinon
                Si (Moy ≥0) et (Moy<5) alors * puisque cette condition est incluse au sinon
                        Ecrire ('Très faible')        on peut écrire uniquement : Si Moy <5
                Sinon
                        Si (Moy < 8) alors
                                Ecrire ('Faible')
                        Sinon
                                Si (Moy < 12) alors
                                        Ecrire ('Bien')
                                Sinon
                                        Si (Moy < 15) alors
                                                Ecrire ('Très bien')
                                        Sinon
                                                Si (Moy ≤ 20) alors
                                                        Ecrire ('Excellent')
                                                Sinon
                                                        Ecrire ('Erreur')
                                                Fsi
                                        Fsi
                                Fsi
                        Fsi
                Fsi
        Fsi
Fin
```

Donc pour les tests imbriqués, on suit la même règle qui dit :

Si le nombre de cas restants ≥ 2 → utiliser le Si

Exemple

```
Lire (N) *   Il existe 100 cas possibles. Donc, on utilise le Si (100 ≥ 2)
Si cond1 alors
        Inst1
Sinon *   nombre de cas restants 99 ≥ 2 → utiliser le Si
        Si cond2 alors
                Inst2
        Sinon *   nombre de cas restants 98 ≥ 2 → utiliser le Si
                Si cond3 alors
```

$Inst_3$

Sinon * nombre de cas restants $97 \geq 2$ → utiliser le Si

Si $cond_4$ **alors**

$Inst_4$

……

Sinon * nombre de cas restants $3 \geq 2$ → utiliser le Si

Si $cond_{98}$ **alors**

$Inst_{98}$

Sinon * nombre de cas restants $2 \geq 2$ → utiliser le Si

Si $cond_{99}$ **alors**

$Inst_{99}$

Sinon * nombre de cas restants 1 → on n'utilise pas le Si

$Inst_{100}$

4. Traduction en Pascal

La structure conditionnelle s'écrit en Pascal comme suit :

If cond **then**

Inst

Else

Inst;

Remarque

S'il y a plus d'une instruction dans *if* ou dans *else*, il faut délimiter les instructions entre *begin* et *end*, pascal exécute une seule instruction par défaut.

Exemple

```
readln (a);
B := 5;
If a > 0 then
B := a + 10;
C := a + 20;
```

Si a=-3, on aura (B= 5 et C = 17) et non pas (B= 7 et C = 17)

On remarque que l'instruction (C:=a+20) a été exécutée malgré que la condition est fausse car cette instruction est en dehors de la condition.

5. Conclusion

Dans ce chapitre nous avons vu les structures conditionnelles ou les structures alternatives ou aussi les tests. On a compris qu'on utilise les tests si le nombre de cas possibles pour une variable est supérieur ou égale à 2. La même règle est valable pour les imbrications c.-à-d. là où on trouve plus de 2 cas on utilise un test.

6. Série d'exercices N°=3

Exercice 3.1

Ecrire un algorithme qui lit la valeur de PH (Potentiel Hydrogène) d'une solution, et affiche si ce milieu est acide, basique ou neutre (Acide : PH < 7, Base : PH > 7, Neutre : PH = 7).

Exercice 3.2

Ecrire un algorithme qui fait lire dans cet ordre : un réel, un opérateur arithmétique (+, - , *, /) et un autre réel. A chaque opérateur valide correspond une opération arithmétique qu'il faut exécuter et afficher le résultat ou un message d'erreur, le cas échéant.

Exercice 3.3

Les formules suivantes donnent la valeur du poids idéal suivant le sexe (H/F), la taille T (cm) et l'âge A (année) :

F1= (3T - 250)*(A+270)/1200 pour les hommes

F2=(T/2 - 30)*(A+180)/200 pour les femmes

Ecrire un algorithme qui calcule le poids idéal ?

Remarque on arrondira au Kilogramme, c'est-à-dire que l'on considèrera que le poids idéal en Kilos est un entier.

Exercice 3.4

Ecrire un algorithme qui calcule la dépense énergétique au repos (DER): pour le métabolisme de base (besoins énergétiques de 24 heures):

Formule de *Harris et Bénédict* :

Pour les hommes :

13.7516 * poids (kg) + 500.33 * taille (m) - 6.7550 * âge (an) + 66.473

Pour les femmes :

9.5634 * poids (kg) + 184.96 * taille (m) - 4.6756 * âge (an) + 655.0955

Exercice 3.5

Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

De 6 à 7 ans → Poussin

De 8 à 9 ans → Pupille

De 10 à 11 ans → Minime

Plus de 12 ans → Cadet

Exercice 3.6

Ecrire un algorithme qui calcule le montant à payer de la quantité Q d'eau consommée. Q est un nombre entier. Le calcul se fait selon le barème suivant :

De 00 à 25 m^3 : 3.60 DA/ m^3

De 26 à 55 m^3 : 11.70 DA/ m^3

De 56 à 82 m^3 : 19.80 DA/ m^3

De 83 et plus : 23.40/ m^3

Remarque un abonné qui consomme une quantité supérieure ou égale à 83 m^3 paiera une partie de la consommation dans chaque tranche.

Exercice 3.7

Ecrire un algorithme qui pour un temps donné (représenté sous forme : heure, minute, seconde) retourne le temps (sous la même représentation) après avoir ajouté une seconde.

Exercice 3.8

Ecrire un algorithme qui pour un numéro de jour, de mois et d'année entrés par clavier détermine s'il s'agit d'une date valide ou non ?

Exemple

12/02/1437 est valide, par contre: 35/12/1200 est non valide

Exercice 3.9

Ecrire un algorithme qui permet de calculer et d'afficher le nombre de jours et de semaines entre deux dates du même mois et de la même année.

Par exemple : il y a 2 semaines et 6 jours entre le 06/02/2016 et le 25/02/2016.

On suppose que l'année introduite par l'utilisateur est toujours non bissextile (il y a toujours 28 jours dans le mois du Février).

Exercice 3.10

Ecrire un algorithme qui résout une équation de deuxième degré : $ax^2+bx+c=0$

Pour des valeurs a, b et c lus au clavier.

Chapitre 4. Les structures de contrôle répétitives

1. Introduction

Ce chapitre expose un autre outil de programmation intéressant qui est les structures répétitives. Elles servent à exécuter des instructions plusieurs fois. Il y a trois modèles de structures itératives, chaque modèle à un usage particulier.

Exercice 1

Ecrire un algorithme qui calcule la moyenne d'un groupe de 3 étudiants ?

Solution

```
Algorithme exo1
Var m1, m2, m3, moy : réel
Début
    Ecrire ('Taper une moyenne')
    Lire (m1)
    Ecrire ('Taper une moyenne')
    Lire (m2)
    Ecrire ('Taper une moyenne')
    Lire (m3)
    moy← (m1 + m2 + m3) / 3
    Ecrire (moy)
Fin
```

Exercice 2

Ecrire un algorithme qui calcule la moyenne d'un groupe de 5 étudiants ?

Solution

```
Algorithme exo2
Var m1, m2, m3, m4, m5, moy : réel
Début
    Ecrire ('Taper une moyenne')
    Lire (m1)
    Ecrire ('Taper une moyenne')
    Lire (m2)
    Ecrire ('Taper une moyenne')
    Lire (m3)
```

```
    Ecrire ('Taper une moyenne')
    Lire (m4)
    Ecrire ('Taper une moyenne')
    Lire (m5)
    moy← (m1 + m2 + m3 + m4 + m5) / 5
    Ecrire (moy)
Fin
```

Exercice 3

Ecrire un algorithme qui calcule la moyenne d'un groupe de 10 étudiants ?

Solution

```
Algorithme exo3
Var m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, moy : réel
Début
    Ecrire ('Taper une moyenne')
    Lire (m1)
    Ecrire ('Taper une moyenne')
    Lire (m2)
    Ecrire ('Taper une moyenne')
    Lire (m3)
    Ecrire ('Taper une moyenne')
    Lire (m4)
    ……
    ……
    Ecrire ('Taper une moyenne')
    Lire (m10)
    moy← (m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9 + m10) / 10
    Ecrire (moy)
Fin
```

Cette solution n'est pas rationnelle! Il faut trouver à un autre outil?

La solution est l'utilisation des : **Boucles**.

Il y a 3 types de boucles : **Pour**, **Tant que** et **répéter**

2. La boucle Pour

2.1 Structure

La boucle pour s'écrit comme suit :

```
Pour i allant de 1 à N faire
    Instructions
Fpour
```

Les instructions entre *Pour* et *Fpour* sont exécutées N fois.

Exercice 4

Ecrire un algorithme qui affiche le mot (Salam) 20 fois.

Solution

```
Algorithme exo4
Var i : entier
Début
    Pour i allant de 1 à 20 faire
        Ecrire ('Salam')
    Fpour
Fin
```

Exercice 5

Ecrire un algorithme qui affiche les 10 premiers nombres.

Solution

```
Algorithme exo5
Var i : entier
Début
    Pour i allant de 1 à 10 faire
        Ecrire (i)
    Fpour
Fin
```

On revient maintenant à notre premier exercice, la solution par les boucles sera comme suit :

```
Algorithme moyenne
Var i : entier
    m, s, moy : réel
Début
    Pour i allant de 1 à 50 faire
        Ecrire ('Taper une moyenne')
        Lire (m)
        s← s + m
    Fpour
    moy← s / 50
    Ecrire (moy)
Fin
```

2.2 Traduction en Pascal

La boucle *Pour* s'écrit en Pascal comme suit :

```
For i := 1 to n do
Begin
    Instructions;
End;
```

Si le nombre d'instructions de la boucle *For* est supérieur à 1 on doit délimiter ces instructions entre *begin* et *end*.

Exercice 6

Traduire l'algorithme précédent en pascal ?

```
Program moyenne;
Var i : integer;
m, S, moy : real;
Begin
        For i := 1 to 50 do
        Begin
                writeln ('Taper une moyenne');
                readln (m);
                S := S+m;
        End;
        Moy := S/50;
        writeln (moy:6:2);
End.
```

Trace d'exécution

Tracer le tableau d'exécution de l'algorithme suivant pour a=12

```
Algorithme addition
Var a, b, i : entier
Début
        1. Lire (a)
        Pour i allant de 1 à 5 faire
            2.  a←a + 3
            3.  b← a + 2
        Fpour
        4. Ecrire (a, b)
Fin
```

Si on numérote les instructions de l'algorithme, les instructions exécutées sont les suivantes :

{1-2-3-2-3-2-3-2-3-2-3-4}

Mémoire			Ecran
i	a	b	
?	12	?	
1	15	17	
2	18	20	
3	21	23	
4	24	26	
5	27	29	27 29

Dans les questions qui demandent la somme ou le nombre, on utilise la règle suivante :

Somme des chiffres : S←S + i

Nombre des chiffres : S←S + 1

Exercice 7

Ecrire un algorithme qui lit un nombre entier et affiche si ce nombre est **premier** ou non. (On dit qu'un nombre est premier si le nombre de ses diviseurs égale à 2)

Exemple 7= {1, 7} : 7 est premier. 9= {1, 3, 9} : 9 n'est pas premier.

Solution

```
Algorithme premier
Var i, X, S : entier
Début
        Ecrire ('Taper un nombre entier')
        Lire (X)
        S← 0
        Pour i allant de 1 à X faire
                Si X mod i = 0 alors
                        S←S + 1
                Fsi
        Fpour
        Si S =2 alors
                Ecrire (S, ' est premier')
        Sinon
                Ecrire (S, ' est non premier')
        Fsi
Fin
```

Remarques

- La liste des diviseurs propres ou stricts ne comporte pas le nombre même.
- Pour optimiser la solution et éviter les itérations inutiles, on travaille avec les diviseurs propres. On remplace X par (X div 2) dans : (**Pour** i allant de 1 à X **faire)** car il n y a aucun diviseur entre la moitié du nombre et le nombre même. Et puisque on n'arrive pas au nombre même dans les diviseurs, le nombre de diviseurs d'un nombre premier sera un seul deviseur qui est le 1, alors (**Si** S =2 **alors**) devient (**Si** S =1 **alors**)

Exercice 8

Ecrire un algorithme qui lit un nombre entier et affiche si ce nombre est **abondant, parfait** ou **déficient**. (Lorsque la somme des diviseurs propres d'un nombre lui est supérieure, le nombre

est dit **abondant**. Lorsque la somme lui est égale, le nombre est dit **parfait**. Lorsque la somme lui est inférieure, le nombre est dit **déficient**.)

Exemple

6 = {1, 2, 3} : 1+2+3 = 6 → 6 = 6, alors 6 est **parfait**

10 = {1, 2, 5} : 1+2+5 = 8 → 8 < 10, alors 10 est **déficient**

12 = {1, 2, 3, 4, 6} : 1+2+3+4+6 = 16 → 16 > 12, alors 12 est **abondant**

Solution

```
Algorithme parfait
Var i, X, S : entier
Début
        Ecrire ('Taper un nombre entier')
        Lire (X)
        S← 0
        Pour i allant de 1 à (X div 2) faire
                Si X mod i = 0 alors
                        S←S + i
                Fsi
        Fpour
        Si S < X alors
                Ecrire (X, ' est déficient')
        Sinon
                Si S > X alors
                        Ecrire (X, ' est abondant')
                Sinon
                        Ecrire (X, ' est parfait')
                Fsi
        Fsi
Fin
```

Exercice 9

Ecrire un algorithme qui lit deux nombres entiers et affiche si ces deux nombres sont amis ou non (on dit que deux nombres sont amis s'ils sont distincts et la somme des diviseurs du premier nombre égale au deuxième et la somme des diviseurs du deuxième nombre égale au premier)

Exemple

24 et 38 :

24= {1, 2, 3, 4, 6, 8, 12} : 1+2+3+4+6+8+12= 36.

38= {1, 2, 19} : 1+2+19=22.

(24≠22) et (38≠36) alors 24 et 38 ne sont pas amis.

220 et 284 :

220 = {1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110} : 1+2+4+5+10+11+20+22+44+55+110= 284.

284 = {1, 2, 4, 71, 142} : 1+2+4+71+142=220.

(220=220) et (284=284) alors 220 et 284 sont amis.

Solution

```
Algorithme amis
Var i, X, Y, S1, S2 : entier
Début
        Ecrire ('Taper un nombre entier')
        Lire (X)
        Ecrire ('Taper un nombre entier')
        Lire (Y)
        S1← 0
        Pour i allant de 1 à (X div 2) faire
                Si X mod i = 0 alors
                        S1←S1 + i
                Fsi
        Fpour
        S2← 0
        Pour i allant de 1 à (Y div 2) faire
                Si Y mod i = 0 alors
                        S2←S2 + i
                Fsi
        Fpour
        Si (S1 =Y) et (S2=X) alors
                Ecrire (X, ' et ', Y, ' sont amis')
        Sinon
                Ecrire (X, ' et ', Y, ' ne sont pas amis')
        Fsi
Fin
```

2.3 Les développements limités

Les exercices des développements limités sont des questions très simples, car il suffit de déduire le terme général de la série et le mettre à l'intérieur d'une boucle dans l'instruction : (S←S + (terme général))

Exercice 10

Ecrire un algorithme qui affiche 10 multiples d'un nombre entier N lu au clavier.

Solution

```
Algorithme exo10
Var i, N, M : entier
Début
    Lire (N)
    Pour i allant de 1 à 10 faire
        M← N * i
        Ecrire (M)
    Fpour
Fin
```

Exercice 11

Ecrire un algorithme qui calcule la somme suivante :

S= 1+1/2+1/3+1/4+……1/N

Solution

```
Algorithme exo11
Var i, N : entier
    S : réel
Début
    Lire (N)
    S←0
    Pour i allant de 1 à N faire
        S← S + 1 / i
    Fpour
    Ecrire (S)
Fin
```

Exercice 12

Ecrire un algorithme qui calcule la somme suivante :

S= 1/2+1/4+1/6+…. +1/2n

Solution

```
Algorithme exo12
Var i, N : entier
    S : réel
Début
    Lire (N)
    S←0
    Pour i allant de 1 à N faire
        S← S + 1 / (2 * i)
    Fpour
    Ecrire (S)
Fin
```

Exercice 13

Ecrire un algorithme qui calcule la somme suivante :

$$S= 1+1/3+1/5+1/7 + \ldots + 1/ 2n+1$$

Solution

```
Algorithme exo13
Var i, N : entier
    S : réel
Début
    Lire (N)
    S←0
    Pour i allant de 1 à N faire
        S← S + 1 / (2 * i + 1)
    Fpour
    Ecrire (S)
Fin
```

Exercice 14

Ecrire un algorithme qui calcule la somme suivante :

$$S=2+4+6+8+\ldots + 2n$$

Solution

```
Algorithme exo14
Var i, N : entier
    S : réel
Début
    Lire (N)
    S←0
    Pour i allant de 1 à N faire
        S← S + 2 * i
    Fpour
    Ecrire (S)
Fin
```

Exercice 15

Ecrire un algorithme qui calcule la somme de la série pour les valeurs de i inferieurs à 100

$$S=2+5+8+11+\ldots +302$$

Solution

```
Algorithme exo15
Var i : entier, S : réel
```

```
Début
    S←2
    Pour i allant de 1 à 100 faire
        S← S + 3 * i + 2
    Fpour
    Ecrire (S)
Fin
```

Exercice 16

Ecrire un algorithme qui calcule la somme de la série pour toutes les valeurs inferieurs à 100

S=2+5+8+11+….+98

Solution

```
Algorithme exo16
Var i : entier
    S : réel
Début
    S←0
    Pour i allant de 1 à 33 faire
        S← S + 3 * i - 1
    Fpour
    Ecrire (S)
Fin
```

Exercice 17

Ecrire un algorithme qui calcule le produit P :

P= 3*4*5*6*7*… * N

Solution

```
Algorithme exo17
Var i, N, P : entier
Début
    Lire (N)
    P←1
    Pour i allant de 1 à N faire
        P← P * (i + 2)
    Fpour
    Ecrire (P)
Fin
```

Exercice 18

Ecrire un algorithme qui calcule la factorielle d'un nombre N strictement positif.

N!=1*2*3*…..*N

Solution

```
Algorithme exo18
Var i, N, F : entier
Début
        Lire (N)
        F←1
        Pour i allant de 1 à N faire
                F← F * i
        Fpour
        Ecrire (F)
Fin
```

Exercice 19

Ecrire un algorithme qui calcule X^n ?

$$2^5 = \underbrace{2*2*2*2*2}_{5 \text{ fois}}$$

Solution

```
Algorithme exo19
Var i, n, P, X : entier
Début
        Lire (X, n)
        P←1
        Pour i allant de 1 à n faire
                P← P * X
        Fpour
        Ecrire (P)
Fin
```

Exercice 20

Ecrire un algorithme qui calcule la somme de la série S :

$$\mathbf{S= X+X^2/2! + X^3/3! + X^4/4!+\ldots + X^n/n!}$$

Solution

```
Algorithme exo20
Var i, N, F, P : entier
    S : réel
```

```
Début
    Lire (N)
    P←1
    F← 1
    S← 0
    Pour i allant de 1 à N faire
        F←F * i
        P← P * X
        S←S + P / F
    Fpour
    Ecrire (S)
Fin
```

Exercice 21

Ecrire un algorithme qui calcule la somme de la série S :

$$\mathbf{S= 1/2+ 2X/3 + 3X^2/4 + 4X^3/5 + \ldots. + nX^{n-1}/n+1}$$

Solution

```
Algorithme exo21
Var i, N, P : entier
    S : réel
Début
    Lire (N)
    P←1
    S←1/2
    Pour i allant de 2 à N faire
        P← P * X
        S←S + (i * P)/(i + 1)
    Fpour
    Ecrire (S)
Fin
```

Exercice 22

Ecrire un algorithme qui calcule la somme de la série S :

$$\mathbf{S= 1/2!+ 2X/3! + 3X^2/4! + 4X^3/5! + \ldots. + (n-1)X^{n-2}/n!}$$

Solution

```
Algorithme exo22
Var i, N, F, P : entier
    S : réel
```

```
Début
    Lire (N)
    P←1
    F← 1
    S← 0
    Pour i allant de 2 à N faire
        F←F * i
        P← P * X
        S←S + (i - 1) * (P / X) / F
    Fpour
    Ecrire (S)
Fin
```

3. La boucle Tant que

3.1 Structure

La boucle *Tant que* s'écrit comme suit :

```
Tant Que (cond) faire
    Instructions
FTQ
```

3.2 Traduction en Pascal

La boucle *Tant que* s'écrit en Pascal comme suit :

```
While (cond) do
Begin
    Instructions;
End;
```

Si le nombre d'instructions dans la boucle *While* est supérieur à 1, on doit délimiter ces instructions entre *begin* et *end*.

Il y a deux types de boucles *Tant Que* (on note la boucle *Tant Que* en algorithme par *TQ*):

3.3 TQ simple (Le nombre de répétions est connu)

C'est une boucle équivalente à la boucle *Pour* sauf que le compteur est manuel.

Structure

La boucle *Tant que* simple s'écrit comme suit :

```
i←1
TQ (i ≤ N) faire
    Instructions
    i←i + 1
FTQ
```

Exercice 23

Ecrire un algorithme qui calcule la moyenne de 100 étudiants (Moy = (D + Exm * 2) / 3)?

Solution

```
Algorithme exo23
Var i : entier
    D, Exm, Moy : réel
Début
    i←1
    TQ i ≤ 100 faire
        Lire (D, Exm)
        Moy←(D + Exm * 2) / 3
        Ecrire (Moy)
        i←i+1
    FTQ
Fin
```

Trace d'exécution

Tracer le tableau d'exécution de l'algorithme suivant pour a=2.5, N=4

```
Algorithme exo23bis
Var i, N : entier
    R, a : réel
Début
    i←1
    Lire (a)
    Lire (N)
    R ←a
    TQ i ≤ N faire
        R ← R * a
        i←i + 1
    FTQ
    Ecrire (R)
Fin
```

Mémoire				Ecran
i	a	R	N	
1	2.5	2.5	4	
2	2.5	6.25	4	
3	2.5	15.62	4	
4	2.5	39.06	4	
5	2.5	97.65	4	97.65

Exercice 24

Ecrire un algorithme qui calcule la moyenne d'un groupe de 100 étudiants?

Solution

```
Algorithme exo24
Var i : entier
    Moy, S, MG : réel
Début
        S←0
        i←1
        TQ i ≤ 100 faire
                Lire (Moy)
                S←S + Moy
                i←i + 1
        FTQ
        MG← S / 100
        Ecrire (MG)
Fin
```

Exercice 25

Traduire l'algorithme de l'exercice précédent en Pascal ?

Solution

```
Program exo25;
Var i : integer;
Moy, S, MG : real;
Begin
        S := 0;
        i := 1;
        While i<=100 do
        Begin
                writeln ('Taper une moyenne');
                readln (Moy);
                S := S+Moy;
                i := i+1;
        End;
        MG := S/100;
        writeln (MG);
End.
```

Exercice 26

Ecrire un algorithme qui lit un nombre entier positif et affiche si ce nombre est **sublime** ou non ? (On dit qu'un nombre est **sublime** si le nombre de ses diviseurs complets et la somme de ses diviseurs complets sont tous deux des nombres parfaits.

Exemple

12 = {1, 2, 3, 4, 6, 12}

Nombre de diviseurs S1= 6 6 est parfait

Somme des diviseurs S2= 28 28 est parfait alors 12 est sublime.

Solution

```
Algorithme sublime
Var i, X, S1, S2, C1, C2 : entier
Début
    Ecrire ('Taper un nombre entier')
    Lire (X)
    S1← 0
    S2← 0
    i←1
    TQ i ≤ X faire
        Si X mod i = 0 alors
            S1←S1 + 1
            S2←S2 + i
        Fsi
        i←i + 1
    FTQ
    C1←0
    i←1
    TQ i ≤ (S1 div 2) faire
        Si S1 mod i = 0 alors
            C1←C1 + i
        Fsi
        i←i + 1
    FTQ
    C2←0
    i←1
    TQ i ≤ (S2 div 2) faire
        Si S2 mod i = 0 alors
            C2←C2 + i
        Fsi
        i←i + 1
    FTQ
    Si (S1=C1) et (S2=C2) alors
        Ecrire (X, ' est sublime')
    Sinon
        Ecrire (X, ' n''est pas sublime')
    Fsi
Fin
```

3.4 TQ à valeur sentinelle (Le nombre de répétions est inconnu)

La caractéristique de cette boucle est que le nombre de répétitions est inconnu préalablement

Structure

La boucle *Tant que* à valeur sentinelle s'écrit comme suit :

```
Lire (X)
TQ (X≠valeur) faire
        Instructions
        Lire (X)
FTQ
```

Exercice 27

Ecrire un algorithme qui lit un ensemble de nombres entiers qui se termine par zéro (0) et calcule la moyenne de ces nombres?

La solution avec la boucle *Pour* n'est pas possible car on connait le début et on ne connait pas la fin (**?**) :

```
Pour i allant de 1 à ? faire
        Lire (N)
        S←S + N
Fpour
M←S / ?
```

La même chose pour la boucle *TQ* du premier type :

```
i←1
Lire (N)
TQ N < ? faire
        Lire (N)
        S←S + N
        i←i+1
FTQ
M←S / ?
```

La solution est impossible car le nombre de répétitions est inconnu.

La solution est dans le deuxième type de *TQ* :

```
Algorithme exo27
Var N, c, S : entier, M : réel
Début
        c←1
        S←0
        Lire (N)
        TQ N≠0 faire
                S←S + N
```

```
        Lire (N)
        c←c + 1
    FTQ
    M←S / c
    Ecrire ('La moyenne = ', M)
Fin
```

Trace d'exécution

Tracer le tableau d'exécution de l'algorithme suivant pour X= {25, -13,-7, 1,-37, 3, 0}

```
Algorithme somme
Var X, S1, S2 : entier
Début
    S1←0
    S2←0
    Lire (X)
    TQ (X≠0) faire
        Si X > 0 alors
            S1←S1 + X
        Sinon
            S2←S2 + X
        Fsi
        Lire (X)
    FTQ
    Ecrire ('La somme des nombres positifs = ', S1)
    Ecrire ('La somme des nombres négatifs = ', S2)
Fin
```

Mémoire			**Ecran**
X	S1	S2	
?	0	0	
25	0	0	
25	25	0	
-13	25	-13	
-7	25	-20	
1	26	-20	
-37	26	-57	
3	29	-57	
0	29	-57	la somme des nombres positifs = 29 la somme des nombres négatifs = -57

Exercice 28

Ecrire un algorithme qui lit une phrase qui se termine par un point, et calcule le nombre de mots? On fait l'hypothèse qu'il n'y a pas de caractères spéciaux et que les mots sont séparés par un et un seul blanc. S'il y a des signes de ponctuation, ils doivent être (comme il se doit) suivis d'un blanc. Il y a au moins un mot qui précède le point.

Solution

```
Algorithme exo28
Var S : entier
    C : caractère
Début
        Lire (C)
        S←0
        TQ C ≠ '.' faire
                Si C=' ' alors
                        S←S + 1
                Fsi
                Lire (C)
        FTQ
        Ecrire ('Le nombre de mots de cette phrase est ', S+1)
Fin
```

Exercice 29

Ecrire un algorithme qui lit une phrase qui se termine par un point, et calcule le nombre de voyelles?

Solution

```
Algorithme exo29
Var S : entier
    C : caractère
Début
        Lire (C)
        S←0
        TQ C ≠ '.' faire
                Si C='a' or C='i' or C='o' or C='u' or C='e' or C='y' alors
                        S←S + 1
                Fsi
                Lire (C)
        FTQ
        Ecrire ('Le nombre de voyelles dans cette phrase est ', S)
Fin
```

Exercice 30

Ecrire un algorithme qui lit une phrase qui se termine par un point, et calcule le nombre de ses caractères ?

Solution

```
Algorithme exo30
Var S, K : entier
    C: caractère
Début
    Lire (C)
    S←0
    K←0
    TQ C ≠ '.' faire
        Si C=' ' alors
            S←S + 1
        Fsi
        K←k + 1
        Lire (C)
    FTQ
    K←K - S
    Ecrire ('Le nombre de caractères est ', K)
Fin
```

Exercice 31

Ecrire un algorithme qui lit une série de nombres qui se termine par 100 et calcule le double de chaque nombre?

Solution

```
Algorithme exo31
Var N, M : entier
Début
    Lire (N)
    TQ N≠100 faire
        M←N * 2
        Ecrire (M)
        Lire (N)
    FTQ
Fin
```

Remarque

On remarque que le dernier nombre (100) n'est pas pris en considération (son double n'est pas affiché) → la solution est dans le troisième type de boucles : la boucle *Répéter*.

4. La boucle Répéter

4.1 Structure

La boucle *Répéter* s'écrit comme suit :

Répéter
    Instructions
Jusqu'à (cond)

4.2 Traduction en Pascal

La boucle *Répéter* s'écrit en Pascal comme suit :

Repeat
    Instructions
Until (cond)

Remarque

Dans la boucle *Repeat*, on n'a pas besoin de *begin* et *end* car les instructions sont délimitées par *Repeat* et *Until*.

4.3 Répéter simple (Le nombre de répétions est connu)

Structure

La boucle *Répéter* simple s'écrit comme suit :

```
i←1
Répéter
    Instructions
    i←i + 1
Jusqu'à i > N
```

Exercice 32

Ecrire un algorithme qui lit un nombre at affiche sa table de multiplication.

Solution

```
Algorithme Table_multiplication
Var i, n, p : entier
Début
    Lire(n)
    i←1
    Répéter
        P ← n*i
        Ecrire (n,' * ', i,' = ', P)
        i←i + 1
    Jusqu'à i > 10
Fin
```

Trace d'exécution

- Tracer le tableau d'exécution de l'algorithme suivant pour N = {12, 23, 11, 13, 20}

- Que fait cet algorithme ?

```
Algorithme Quoi
Var i, N, M : entier
Début
        i←1
        Répéter
                Lire(N)
                Si N mod 2 = 0 alors
                        M ← N div 2
                Sinon
                        M ← N*2
                Fsi
                Ecrire (M)
                i←i + 1
        Jusqu'à i > 5
Fin
```

Mémoire			Ecran
i	N	M	
1	12	6	6
2	23	46	46
3	11	22	22
4	13	26	26
5	20	10	10
6	20	10	

Cet algorithme lit 5 nombres entiers et teste chaque nombre. S'il est pair, il affiche sa moitié.

S'il est impair, il affiche son double.

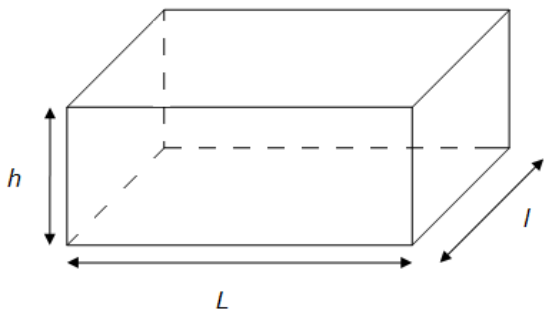
Exercice 33

Ecrire un algorithme qui lit les dimensions de 10 parallélépipèdes, at calcule leurs volumes.

Sachant que V=L*I*H, tel que : L : longueur de la base, I : largeur de la base, H : hauteur.

Solution

```
Algorithme parallélépipède
Var j : entier
    L, I, H, V : réel
```



```
Début
    j←1
    Répéter
        Lire (L, I, H)
        V ← L * I * H
        Ecrire ('Le volume = ', V)
        j←j + 1
    Jusqu'à j > 10
Fin
```

Exercice 34

Traduire l'algorithme de l'exercice précédent en Pascal ?

Solution

```
Program exo34;
Var j : integer;
L, I, H, V : real;
Begin
    j:=1;
    repeat
        writeln ('Taper les dimensions du parallélépipède');
        readln (L, I, H);
        V := L*I*H;
        writeln ('Le volume = ', V);
        j:=j+1;
    until j>10;
End.
```

Exercice 35

Écrire un algorithme qui affiche la valeur de la fonction x^2 pour les dix premiers entiers positifs.

Solution

```
Algorithme exo35
Var i, P : entier
Début
    i←1
    Répéter
        P←i * i
        Ecrire ('Le carrée de ', i, ' est ', P)
        i←i + 1
    Jusqu'à i > 10
Fin
```

4.4 Répéter à valeur sentinelle (Le nombre de répétions est inconnu)

Structure

La boucle *Répéter* à valeur sentinelle s'écrit comme suit :

```
Lire (X)
Répéter
    Instructions
    Lire (X)
Jusqu'à X=valeur
```

On utilise cette structure si la dernière valeur de la série n'est pas concernée par le traitement, par contre si la dernière valeur est prise en considération on utilise la structure suivante:

```
Répéter
    Lire (X)
    Instructions
Jusqu'à X=valeur
```

Exercice 36

Ecrire un algorithme qui lit une série de nombres qui se termine par 0, cette série représente des mesures en pouces, et convertit chaque mesure en centimètre.

Solution

```
Algorithme conversion
Var   C, P : réel
Début
    Lire (P)
    Répéter
        C←P * 2.54
        Ecrire (C)
        Ecrire (P, ' = ', C, ' cm')
        Lire(P)
    Jusqu'à P=0
Fin
```

Trace d'exécution

Tracer le tableau d'exécution de l'algorithme suivant pour a = {25, 10, -13, 30, 80, -36, 1000}

```
Algorithme inconnu
Var   a : entier
      r, s : réel
Début
    Lire (a)
    s←0
    Répéter
        Si a ≥ 0 alors
```

```
            r←racine (a)
            Ecrire (r)
            s←s + r
        Fsi
        Lire (a)
    Jusqu'à a=1000
    Ecrire (s)
Fin
```

Mémoire			Ecran
a	r	s	
25	?	0	
25	5	5	5
10	3.16	8.16	3.16
-13	3.16	8.16	
30	5.47	13.63	5.47
80	8.94	22.57	8.94
-36	8.94	22.57	
1000	8.94	22.57	22.57

Exercice 37

Ecrire un algorithme qui calcule la conversion en dinars d'un ensemble de sommes en euros saisies au clavier. L'ensemble se termine par 0. Le taux de change doit être lu aussi.

Solution

```
Algorithme exo37
Var   Som, T, D : réel
Début
    Lire (som)
    Lire (T)
    Répéter
        D←som * T
        Ecrire (som,' euros = ', D,' dinars')
        Lire (som)
    Jusqu'à som=0
Fin
```

Exercice 38

Ecrire un algorithme qui lit les noms des candidats désirants participer à un concours et s'arrête lorsqu'il rencontre deux noms consécutifs identiques. L'algorithme doit afficher à la fin le nombre de participants.

Solution

```
Algorithme exo38
Var   C : entier, nom1, nom2 : chaine de caractères
```

```
Début
    C←0
    Lire (nom1)
    Répéter
        C←C + 1
        nom2←nom1
        Lire (nom1)
    Jusqu'à nom1=nom2
    Ecrire ('Le nombre de participants est ', C)
Fin
```

4.5 Répéter pour le contrôle de saisie

Parmi les utilisations uniques et spécifiques de la boucle *Répéter*, la validation des entrées ou le contrôle de saisie; c.-à-d. la vérification des données avant de les manipuler s'il y a des conditions.

Exercice 39

Ecrire un algorithme qui lit une moyenne d'un étudiant et affiche s'il est Admis ou Ajourné ? Quand on lit une moyenne, elle peut être non valide (exemple : Moy = -12, ou Moy =31), alors il faut tester cette valeur avant de la manipuler:

```
Lire (Moy)
Si (Moy ≤ 0) ou (Moy ≥ 20) alors
    Lire (Moy)
    Si (Moy ≤ 0) ou (Moy ≥ 20) alors
        Lire (Moy)
        Si (Moy ≤ 0) ou (Moy ≥ 20) alors
            Lire (Moy)
            …….
            …….
```

Mais, combien de fois on doit tester la justesse de la moyenne ? Le nombre de frappes où la valeur est fausse est inconnu, donc la solution est dans l'utilisation de la boucle *Répéter* de contrôle de saisie :

Solution

```
Algorithme exo39
Var Moy : réel
Début
    Répéter
        Lire (Moy)
    Jusqu'à (Moy ≥ 0) et (Moy ≤ 20)
    Si Moy ≥ 10 alors
        Ecrire ('Admis')
```

```
        Sinon
            Ecrire ('Ajourné')
        Fsi
Fin
```

Trace d'exécution

Tracer le tableau d'exécution de l'algorithme suivant pour âge = {-12, -20, 17, 25, -33, -50, 63, 2, 0, 6}

```
Algorithme categorie
Var i, age : entier
Début
        Pour i allant de 1 à 5 faire
            Répéter
                Lire (age)
            Jusqu'à (age > 0)
            Si age < 12 alors
                Ecrire ('Enfant')
            Sinon
                Si age < 18 alors
                    Ecrire ('Adolescent')
                Sinon
                    Ecrire ('Adulte')
                Fsi
            Fsi
        Fpour
Fin
```

Mémoire		Ecran
i	age	
1	-12	
1	-20	
1	17	Adolescent
2	25	Adulte
3	-33	
3	-50	
3	63	Adulte
4	2	Enfant
5	0	
5	6	Enfant

Exercice 40

Ecrire un algorithme qui lit une série de distances qui se termine par 0 et calcule leurs moyennes ?

Solution

```
Algorithme exo40
Var C : entier
D, S, Moy : réel
Début
        S←0
        C←0
        Répéter
                Lire (d)
        Jusqu'à d ≥ 0
        Répéter
                S←S + d
                C←C + 1
                Répéter
                        Lire (d)
                Jusqu'à d ≥ 0
        Jusqu'à d=0
        Moy←S / C
        Ecrire ('La moyenne de ces distances est ', Moy)
Fin
```

5. Comparaison entre les différentes boucles

Comparaison entre Pour et TQ-Répéter

boucle / Différence	*Pour*	*TQ/Répéter*
Compteur	Automatique	Manuel
Nombre de répétitions	Connu	Connu/inconnu

Comparaison entre TQ et Répéter

boucle / Différence	*TQ*	*Répéter*
Condition	d'entrée	de sortie
Nombre d'exécutions	0 et plus	1 et plus

6. Conclusion

Dans ce dernier chapitre, nous avons vu les structures répétitives ou les itérations. Leur utilisation est indispensable pour un nombre élevé des données ou un travail répétitif. Nous avons vu les trois boucles : *Pour*, *Tant que*, et *Répéter*. On a pu distinguer la différence entre les différentes boucles.

7. Série d'exercices N°=4

Exercice 4.1

Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple si on lit 6, l'algorithme calcule : 1+2+3+4+5+6 = 21.

Exercice 4.2

Ecrire un algorithme qui calcule la somme des carrées des entiers de 1 à 10.

Exercice 4.3

Ecrire un algorithme qui lit un nombre réel X et un nombre entier N et calcule leur produit en n'utilisant que l'addition.

Exercice 4.4

Ecrire un algorithme qui demande un nombre compris entre 50 et 100, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 50, on fera apparaitre un message 'Plus petit', et inversement 'Plus grand' si le nombre est inférieur à 50.

Exercice 4.5

Ecrire un algorithme qui calcule la température moyenne du mois de Janvier à Sétif sachant que les températures dans ce mois ne descendent jamais au-dessous de -20°c et ne monte jamais au-dessus de 30°c.

Exercice 4.6

Ecrire un algorithme qui affiche les diviseurs communs de 2 nombres X et Y (X et Y > 0)

Exercice 4.7

Ecrire un algorithme qui calcule la factorielle d'un nombre N entier (N peut être positif, négatif ou nul).

Exercice 4.8

Ecrire un algorithme qui calcule la somme factorielle de 1 à N, N : nombre lu au clavier.

Exemple N=5, S=1! + 2 !+ 3!+ 4!+ 5!

Exercice 4.9

Ecrire un algorithme qui affiche les termes de la suite de *Fibonacci* de 0 à 20. Cette suite est définie comme suit : F(0)=1, F(1)=1, F(n)= F(n-2)+ F(n-1) pour tout $n \geq 2$.

Exercice 4.10

Ecrire un algorithme qui calcule le plus petit commun multiple (PPCM) de deux nombres entiers N et M. vérifier par programme, la relation : PPCM (N,M) * PGCD (N,M) = N*M.

Exercice 4.11

Ecrire un algorithme qui lit une phrase qui se termine par un point et affiche le mot le plus long.

Exercice 4.12

Ecrire un algorithme qui convertit un nombre décimal N en binaire (N>0). Pour convertir un nombre, on le divise par 2 et on affiche le reste de la division. Cette opération se répètera jusqu'à ce que le résultat de la division égale à 0.

Exercice 4.13

La suite de *Syracuse*$_p$ avec p > 1 est définie de la façon suivante :

$a_0 = p$

$a_{n+1} = a_n/2$ si a_n est pair

$a_{n+1} = 3 * a_n + 1$ si a_n est impair

Ecrire un algorithme qui calcule la valeur du nième terme de la suite *Syracuse*$_p$.

Chapitre 5. Mini Projet : Calcul de Zakat

Dans ce projet, on veut calculer Zakat des biens suivants : bétail, produits agricoles, Or, argent, numéraires. Les détails de calcul de chaque type sont présentés comme suit :

1. Bétail

1/chameaux :

Quorum		Quote-part
de	à	
5	24	Pour 5 chameaux un C
25	35	M
36	45	L
46	60	H
61	75	J
76	90	2L
91	120	2H

Règle1 : Au-delà de 120 chameaux pour chaque 50 têtes → 1H et pour chaque 40 têtes →1L

Exemple : 130 = 2*40+1*50 = 2L+1H

140 = 1*40+2*50 = 1L +2H

150 = 3*50 = 3H

…etc.

Remarque : La même quote-part reste valable pour les valeurs comprises dans l'intervalle

[130-140[→ 2L+1H

[140-150[→ 1L+2H,…

2/bovins :

Quorum		Quote-part
de	à	
30	39	T
40	59	Mo
60	69	2T
70	79	Mo + T
80	89	2 Mo
90	99	3T
100	119	Mo + 2T
120	129	3Mo ou 4T

Règle2 : Au-delà de 120 bovins pour chaque 40 têtes→ 1Mo et pour chaque 30 têtes→1T.

3/ovins :

Quorum		Quote-part
de	à	
40	120	1C
121	200	2C
201	399	3C

Règle3 : Au-delà de 400 ovins, pour 100 têtes →1C

2. Produits agricoles

En cas de :

Irrigation naturelle → quote-part=10%

Irrigation artificielle→ quote-part=5%

Remarque : Pour le blé la règle est valable au-delà de 675Kg

3. L'or et l'argent

1/ l'argent : le quorum au-delà de 595g

2/ l'or : le quorum :

. Au-delà de 85g pour l'or de 24 carats

. Au-delà de 97g pour l'or de 21 carats

. Au-delà de 113g pour l'or de18 carats

La quote-part =2.5%=1/40

4. Numéraires

L'équivalent de 85g d'or, au cours actuel le quorum =395250 DA, la quote-part=2.5%

Question

Ecrire un algorithme pour chaque type de bien qui calcule " Zakat" de façon automatique, Remarquer que, dans le type d'animaux, qu'il existe une partie déclarative et une partie avec règles.

Conclusion

Ce manuscrit a porté sur l'algorithmiques dans sa dimension conceptuelle et dans sa dimension méthodique. On a commencé par l'analyse des problèmes, étape primordiale avant l'écriture de l'algorithme, elle sert à déterminer la liste des objets de travail, ainsi leurs types. L'algorithme en lui-même est un enchainement logique des instructions pour la résolution des problèmes. Pour élargir l'ensemble des problèmes qui peuvent être résolus, on a vu des structures de données comme les structures conditionnelles ou les tests pour répondre aux cas où il y a plusieurs possibilités de solution, et les structures répétitives ou les itérations pour répondre aux cas où il y a un grand nombre de données, ou un travail répétitif.

Toute la terminologie algorithmique utilisée a été traduit en langage Pascal, un langage pédagogique et proche de l'algorithmique.

Tous les concepts expliqués étaient appuyés par des exemples, plus de 100 exercices dont près de 60 exercices résolus ont accompagnés les explications des nouvelles notions.

Le manuel s'achève par un projet récapitulatif englobant toutes les informations vues dans ce cours.

Références

[1] *Byron S. Gottfried, Programmation Pascal avec Turbo Pascal cours et problèmes, série Schaum, 2e édition, France, 1995*.

[2] Equipe de formation, *Informatique 1*, support de cours, département d'enseignement de base en Technologie, Université de Sétif, 2005 à 2016.

[3] Benyoucef Mourad., *Structures de données abstraites*, support de cours, département d'informatique, Université de Sétif, 1996.

[4] Edouard Thiel, *Algorithmes et programmation en Pascal*, support de cours, Faculté des Sciences de Luminy, 2004.

[5] http://www.calculateur.com/ Consulté en Avril 2016.