Algorithmique & Programmation

I-Les structures de données

I.1) Déclaration des constantes

En Algorithmique :

Tableau de Déclaration des Objets

	Objets	Type/Nature	Rôle
Général	Nom	Constante = valeur de la constante	Rôle
Exemples	Annee G Auteur Existe	Constante = 2011 Constante = 9.81 Constante = "Chebbi" Constante = Vrai	

En Pascal :

- ***Syntaxe :*** **CONST <nom_constante> = <valeur_constante> ;**
- ***Exemples :***

```
CONST annee = 2011 ;
      g = 9.81 ;
      auteur = 'Chebbi' ;
      existe = True ;
```

I.2) Déclaration des variables

En Algorithmique :

T.D.O

	Objets	Type/Nature	Rôle
Général	Nom	Type de la variable	Rôle
Exemples	Heure Moy Phrase	Entier Réel Chaîne de caractères	

En Pascal :

- ***Syntaxe :*** **VAR <nom_variable> : type_variable ;**
- ***Exemples :***

```
VAR Heure : Integer ;
    Moy : Real ;
    Phrase : String ;
```

I.3) Déclaration d'un tableau à une dimension

Première formulation

EN ALGORITHMIQUE

T.D.O.

Objet	Type/Nature	Rôle
Ident_tableau	Tableau de taille et de type_éléments	
MOY	Tableau de 30 Réels	

EN PASCAL

```
VAR
  Ident_tableau : ARRAY [Binf..Bsup] OF
                                  Type_éléments ;

VAR   Moy : array [1..30] of real ;
```

Deuxième formulation

EN ALGORITHMIQUE

Tableau de déclarations des nouveaux types

Type
Nom_type = tableau de taille et de type_éléments

T.D.O.

Objet	Type/Nature	Rôle
Ident_tableau	Nom_type	

EN PASCAL

```
TYPE
     Nom_type = ARRAY [Binf..Bsup] OF
                                  Type_éléments ;
VAR
     Ident_tableau : Nom_type ;
```

Exemple :

```
Type  Tab = Array [1..100] of string ;
Var   T : Tab ;
```

I.4) Le type Enuméré

En algorithmique :

Tableau de déclaration des nouveaux types

Type
Nom_du_type = (valeur1, valeur2, valeur3, …)

En pascal :

TYPE Nom_du_type = (valeur1, valeur2, valeur3, …) ;

Exemples : **TYPE** SEMAINE = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche) ;
MARQUE = (renault, citroen, peugeot, ford, toyota) ;

I.5) Le type Intervalle

En algorithmique :

Tableau de déclaration des nouveaux types

Type
Nom_du_type = borne inférieur . . borne supérieur

En pascal :

TYPE Nom_du_type = borne inférieur . . borne supérieur ;

Exemples : **TYPE** SEMAINE = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche) ;
JOUR_TRAVAIL = lundi . . vendredi ;
MOIS = 1 . . 12 ;
Minuscules = 'a' . . 'z' ;

II- Les structures simples

II.1) L'opération d'entrée (lecture, saisie)

En Analyse	En Algorithmique	En Pascal
Variable = Donnée ("Message")	Ecrire("Message") ; Lire(variable)	Write('Message') ; Readln(variable) ;

II.2) L'opération de sortie (écriture, affichage)

	Analyse / Algorithme	Pascal	Résultat sur l'écran
Affichage d'un texte	**Ecrire** ("Bonjour")	**WriteLn** ('Bonjour') ;	Bonjour
Affichage du contenu d'une variable (soit A = 4 et B = 8)	**Ecrire** (A) **Ecrire** (A, B) **Ecrire** (A, " ", B)	**WriteLn** (A) ; **WriteLn** (A, B) ; **Writeln** (A, ' ' , B) ;	4 48 4 8
Affichage de la valeur d'une expression	**Ecrire** (2 * B DIV 3) **Ecrire** (A>B)	**Writeln** (2 * B DIV 3) ; **Write** (A > B) ;	5 False
Affichage mixte (textes et variables)	**Ecrire** ("La valeur est : ", B) **Ecrire** (A, "+", B, "=", a+b)	**Writeln** ('La valeur est : ', B) ; **Write** (A, '+', B, '=', a+b) ;	La valeur est : 8 4 + 8 = 12

II.3) L'opération d'affectation

Analyse / Algorithme	Pascal
Variable ⇦ **Valeur**	Variable **:=** Valeur ;

Exemples

Analyse et Algorithme	Pascal	Commentaire	Résultat
A ⇦ 5	A := 5 ;	La variable A reçoit la valeur 5	A = 5
B ⇦ A + 3	B := A + 3 ;	B reçoit la valeur de l'expression A+3	B = 8
A ⇦ A +1	A := A +1 ;	A reçoit sa valeur actuelle incrémentée de 1	A = 6
C ⇦ long("lycée")/2	C := length('lycée')/2 ;	C reçoit la valeur de l'expression …	C = 2.5

III–Les structures de contrôles conditionnelles

III.1) La structure de contrôle conditionnelle simple

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s)] **Si** condition(s) **Alors** Traitement 1 **Sinon** Traitement 2 **Fin Si**	initialisation(s) **Si** condition(s) **Alors** Traitement 1 **Sinon** Traitement 2 **Fin Si**	..... ; {initialisation(s)} **IF** condition(s) **THEN Begin** Traitement 1 ; **End** **ELSE Begin** Traitement 2 ; **End ;**

III.2) La structure de contrôle conditionnelle généralisée

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s)] **Si** condition1 **Alors** Trait1 **Sinon Si** condition2 **Alors** Trait2 **Sinon Si** condition3 **Alors** Trait3 **Sinon Si** …… **Sinon Si** condition n-1 **Alors** Trait n-1 **Sinon** Trait n **Fin Si**	initialisation(s) **Si** condition1 **Alors** Trait1 **Sinon Si** condition2 **Alors** Trait2 **Sinon Si** condition3 **Alors** Trait3 **Sinon Si** …… **Sinon Si** condition n-1 **Alors** Trait n-1 **Sinon** Trait n **Fin Si**	..... ; {initialisation(s)} **IF** condition1 **THEN Begin** Trait1; **End** **ELSE IF** condition2 **THEN Begin** Trait2; **End** **ELSE IF** …… **ELSE IF** condition n-1 **THEN Begin** Trait n-1; **End** **ELSE Begin** Trait n; **End ;**

III.2) La structure de contrôle conditionnelle à choix multiples

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s)] **Selon** sélecteur **Faire** valeur1 : trait1 valeur2 : trait2 valeur5, valeur8 : trait3 valeur10..valeur30 : trait4 ... valeur n-1 : trait n-1 **Sinon** trait n **Fin Selon**	initialisation(s) **Selon** sélecteur **Faire** valeur1 : trait1 valeur2 : trait2 valeur5, valeur8 : trait3 valeur10..valeur30 : trait4 ... valeur n-1 : trait n-1 **Sinon** trait n **Fin Selon**	.....; {initialisation(s)} **CASE** sélecteur **OF** valeur1 : trait1 ; valeur2 : trait2 ; valeur 5, valeur 8 : trait3 ; valeur 10..valeur 30 : trait4 ; ... valeur n-1 : trait n-1 ; **Else** trait n ; **END ;**

IV– Les structures de contrôles itératives

IV.1) La structure de contrôle itérative complète

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s)] **POUR** Cp **de** Vi **à** Vf **FAIRE** Traitement **FinPour**	initialisation(s) **POUR** Cp **de** Vi **à** Vf **FAIRE** Traitement **FinPour**	..... ; { initialisation(s) } **FOR** Cp:=Vi **TO/DOWNTO** Vf **DO** **Begin** Traitement ; **End** ; *{**TO** lorsque Vi ≤ Vf}* *{**DOWNTO** lorsque Vi ≥ Vf}*

IV.2) La structure de contrôle itérative à condition d'arrêt

a.Première formulation :

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s)] **REPETER** Traitement **JUSQU'A** condition(s)	initialisation(s) **REPETER** Traitement **JUSQU'A** condition(s) *{jusqu'à condition soit vraie}*	.....; {initialisation(s)} **REPEAT** Traitement ; **UNTIL** condition(s) **;**

b.Deuxième formulation :

En Analyse	En Algorithmique	En Pascal
Nom_objet = [initialisation(s) **TANT QUE** condition(s) **FAIRE** Traitement **Fin Tant que**	initialisation(s) **TANT QUE** condition(s) **FAIRE** Traitement **Fin Tant que** *{Tant que condition est vraie répéter le traitement}*	.....; {initialisation(s)} **WHILE** condition(s) **DO** **Begin** Traitement ; **End** ;

V–Saisie contrôlée

Exemple1 : Saisir un entier N tel que (20 ≤ N ≤ 100)

Réponse 1	Réponse 2
REPEAT Writeln ('Donner un entier') ; **Readln** (N) ; **UNTIL** (20 <= N) and (N<=100) **;**	**REPEAT** Writeln ('Donner un entier') ; **Readln** (N) ; **UNTIL** N IN [20..100] **;**

N.B : Réponse 2, est valable que si N est un entier dans [0..255]

Exemple2 : Saisir un entier M impair

Réponse 1	Réponse 2
REPEAT Writeln ('Donner un entier') ; **Readln** (M) ; **UNTIL** M mod 2 <> 0 **;**	**REPEAT** Writeln ('Donner un entier') ; **Readln** (M) ; **UNTIL** ODD (M)**;**

Exemple3 : Saisir un réel R tel que (-10 ≤ R ≤ 10)

Réponse 1	Réponse 2 (incorrecte)
REPEAT Writeln('Donner un réel') ; **Readln**(R) ; **UNTIL** (-10 <= R) and (R<=10) **;**	**REPEAT** Writeln('Donner un réel') ; **Readln**(R) ; **UNTIL** R IN [-10..10] **;**

Exemple4 : Saisir une lettre L alphabétique

Réponse 1	Réponse 2
REPEAT Writeln ('Donner une lettre alphabétique') ; **Readln** (L) ; **UNTIL** ('A'<=Upcase(L)) and (Upcase(L)<='Z') **;**	**REPEAT** Writeln ('Donner une lettre alphabétique') ; **Readln** (L) ; **UNTIL** Upcase(L) IN ['A'..'Z']**;**

Exemple5 : Saisir une chaîne de caractère en majuscule

Réponse1	Réponse2
REPEAT Writeln ('Donner une chaîne en majuscule') ; **Readln** (ch) ; verif := true ; i := 0 ; **Repeat** i := i+1 ; If not (ch[i] in ['A'..'Z']) then verif := false ; **Until** (verif = false) or (i = length(ch)) ; **UNTIL** verif = true ;	**REPEAT** Writeln ('Donner une chaîne en majuscule') ; **Readln** (ch) ; i := 0 ; **Repeat** i := i+1 ; Verif := ch[i] in ['A'..'Z'] ; **Until** (not verif) or (i = length(ch)) ; **UNTIL** verif ;

Exemple6 : Remplir un tableau T par n entiers positifs non nuls

Réponse

```
FOR i :=1 TO n DO
      REPEAT
          Writeln ('Introduire le contenu de la case ', i) ;
          Readln (T[i]) ;
      UNTIL T[i] > 0 ;
```

VI- Les sous programmes

VI.1) Les procédures

- Une procédure est un **sous-programme** qui permet la résolution d'un sous-problème précis et qui peut *transmettre zéro, un ou plusieurs résultats* au programme appelant.

Exemple1 : Procédure qui fait la permutation de deux variables entières.

```
Procedure PERMUT (VAR x, y : Integer) ;
Var aux : integer ;
Begin
    Aux := X ;
    X := Y ;
    Y := Aux ;
End ;
```

Exemple2 : Procédure qui fait la saisie d'un tableau T par n réels compris entre 0 et 20.

```
Procedure SAISIE_T (n : Integer ; VAR T : TAB) ;
Var i : Integer ;
Begin
  FOR i :=1 TO n DO
        REPEAT
          Writeln ('Introduire le contenu de la case ', i) ;
          Readln (T[i]) ;
        UNTIL (0 <= T[i]) and (T[i] <= 20) ;
End ;
```

Exemple3 : Procédure qui fait l'affichage d'un tableau T de n éléments

```
Procedure AFFICHE_T (n : Integer ; T : TAB) ;
Var i : Integer ;
Begin
  FOR i :=1 TO n DO Write (T[i], ' ') ;
End ;
```

- **L'entête de la définition :**
  - ***En analyse et algorithmique :***
    DEF PROC Nom_procédure (pf_1 :$type_1$; pf_2 :$type_2$; … ; pf_n :$type_n$)
  - ***En Pascal :***
    PROCEDURE Nom_procédure (pf_1 :$type_1$; pf_2 :$type_2$; … ; pf_n :$type_n$) ;

- **L'appel :**
  - ***En analyse et algorithmique :***
    PROC Nom_procédure (pe_1 , pe_2 , … , pe_n)
  - ***En Pascal :***
    Nom_procédure (pe_1 , pe_2 , … , pe_n) ;

VI.2) Les fonctions

- Une fonction est un **sous-programme** qui permet la résolution d'un sous-problème précis et doit retourner (renvoyer) un **seul résultat** de type simple *(entier, caractère, réel, booléen ou chaîne)* au programme appelant. Il s'agit d'un cas particulier des procédures.

Exemple1 : Fonction qui détermine la plus petite valeur dans un tableau T de n réels.

```
Function MINIMUM (n : Integer ; T : TAB) : Real ;
Var i : Integer ;
      Min : Real ;
Begin
  Min := T[1] ;
  FOR i :=2 TO n DO  If (T[i] < min) Then min := T[i] ;
  Minimum := min ;
End ;
```

- **L'entête de la définition :**
  - ***En analyse et algorithmique :***
    DEF FN Nom_fonction (pf_1 :$type_1$; pf_2 :$type_2$; … ; pf_n :$type_n$) : Type_résultat
  - ***En Pascal :***
    Function Nom_fonction (pf_1 :$type_1$; pf_2 :$type_2$; … ; pf_n :$type_n$) : Type_résultat ;

- **L'appel :**
  - ***En analyse et algorithmique :***
    Nom_objet ⇦ **FN** Nom_fonction (pe_1 , pe_2 , … , pe_n)
  - ***En Pascal :***
    Nom_objet := Nom_fonction (pe_1 , pe_2 , … , pe_n) ;

STRUCTURE GENERALE D'UN PROGRAMME PASCAL

PROGRAM Nom_programme ; {En-tête du programme}*

Uses ... ; {Utilisation des unités / bibliothèques}*

Const ... ; {Déclaration des constantes}*

Type ... ; {Déclaration des types}*

Var ... ; {Déclaration des variables}*

*{============ Définition des procédures ==================}**

Procedure Nom_procédure (paramètres formels) ;

{Déclarations locales : Const, Type, Var, Function, Procedure, ...}*

Begin

Instructions de la procédure ;

End ;

*{============ Définition des fonctions ==================}**

Function Nom_fonction (paramètres formels) : Type_résultat ;

{Déclarations locales : Const, Type, Var, Function, Procedure, ...}*

Begin

Instructions de la fonction ;

Nom_fonction := résultat ;

End ;

*{======================== **P. P.** ========================}*

BEGIN {Début du programme principal}

Instructions ;

…… ;

{Bloc principal du programme avec appel des procédures et des fonctions}

END. {Fin du programme}

*** : facultatif**