Algorithmique

CHAPITRE 8 : LES FONCTIONS ET PROCEDURES

Sommaire

1. Définitions
2. Notation des fonctions
3. Exemples utilisant les fonctions
4. Notation des procédures
5. Exemples utilisant les procédures
6. Exercices
7. Corrections
8. Projet : calcul de moyenne
9. Correction projet

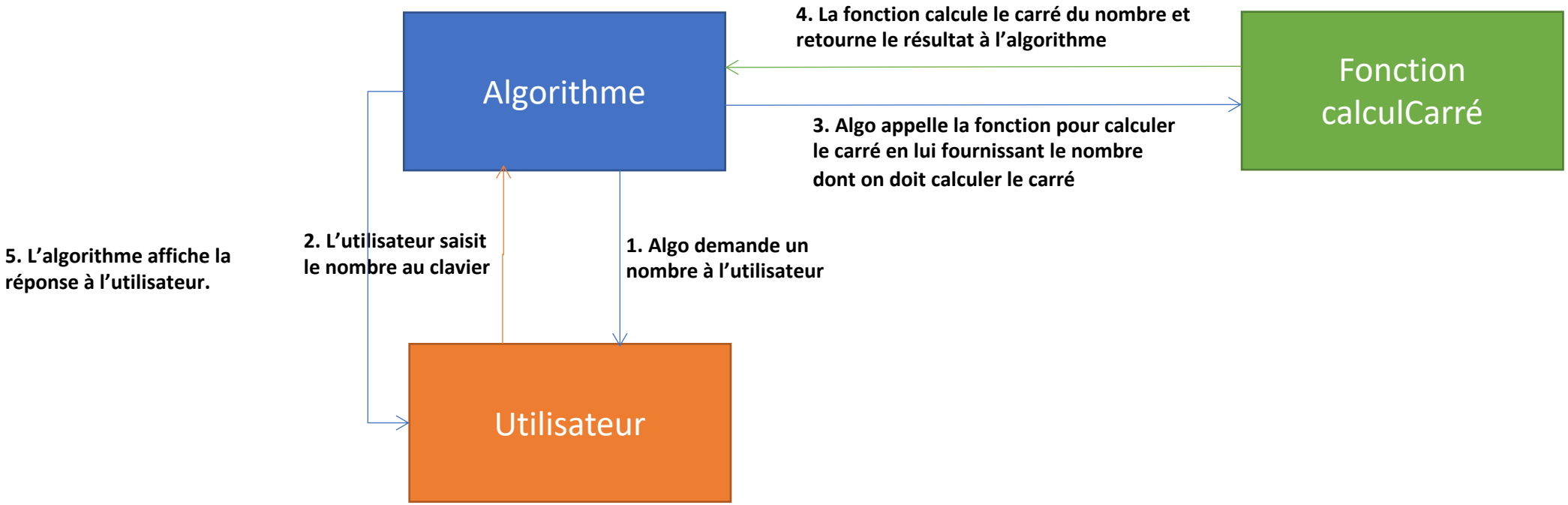
1. Définitions

- Supposons les sujets suivants:
  - Ecrire un algorithme qui demande le nom de l'utilisateur et son âge et qui affiche le message « Hello nom_de_l'utilisateur vous avez age_de_l'utilisateur »
  - Ecrire un algorithme qui demande le nom de l'utilisateur et affiche le message « Hello nom_de_l'utilisateur »
- Si on devait écrire ces algorithmes, on allait pour chacun d'eux écrire des instructions pour demander à l'utilisateur son nom.
- On répète deux fois le code pour demander le nom de l'utilisateur. Pour éviter cela on fait appel à la notion de **fonction** ou **procédure.**
- Une fonction ou procédure permet de créer des blocs d'instructions autonomes qui peuvent être appelés depuis divers endroits d'un algorithme.
- Les **fonctions et procédures** sont identifiées par des étiquettes grâce auxquelles on peut les appeler dans un algorithme.
- De façon imagée, les procédures et fonctions sont comme des algorithmes mais la différence ici est que ce sont des algorithmes désignés par un nom et qui peuvent être appelés dans un autre algorithme.
- Une procédure/fonction peut appeler une autre procédure et fonction lors de son exécution.

1. Définitions

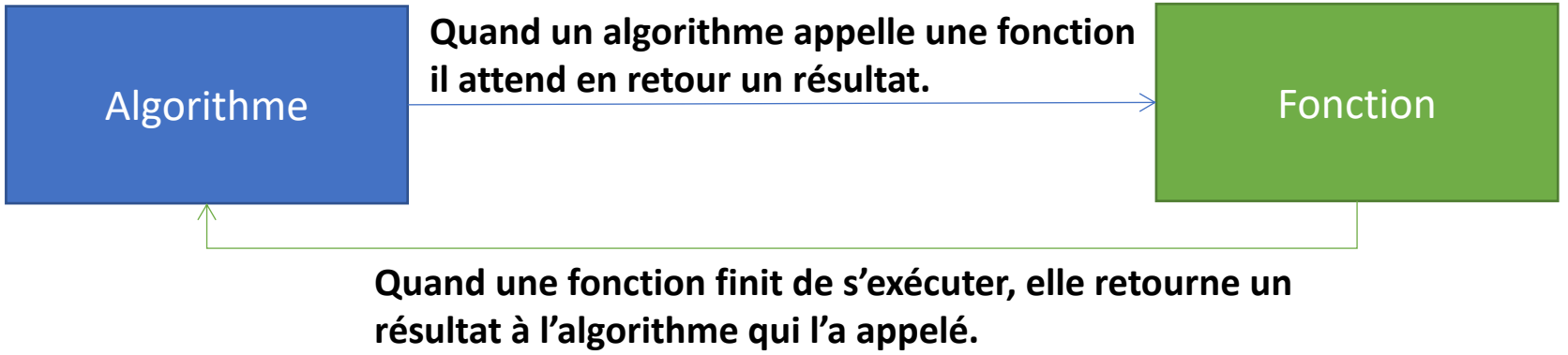
- Il existe une différence entre une procédure et une fonction:
  - Une fonction lorsqu'elle est appelée, exécute son bloc d'instructions mais à la fin renvoie un résultat.
  - Une procédure quant à elle s'exécute sans retour de résultat.

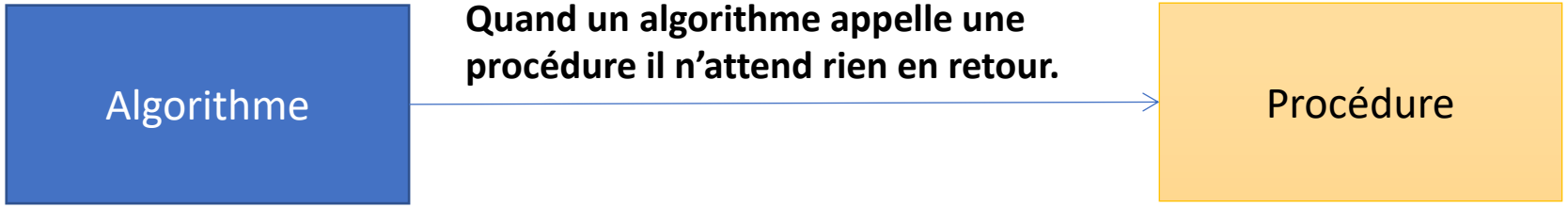
- Une fonction comme une procédure peut avoir besoin d'informations pour pouvoir s'exécuter, on parle de **paramètres.** Par exemple si on crée une fonction **calculCarré** qui doit calculer le carré d'un nombre, la fonction pour s'exécuter à besoin de connaitre le nombre dont elle doit calculer le carré. Ce nombre lui est fourni par l'algorithme qui l'appelle. Ce nombre est appelé **paramètre**.



1. Définitions

- Dans l'exemple précédent si on utilisait une procédure, c'est elle qui allait se charger d'afficher le résultat de l'opération à l'utilisateur vu qu'une procédure par définition ne renvoie pas de résultat après son traitement.



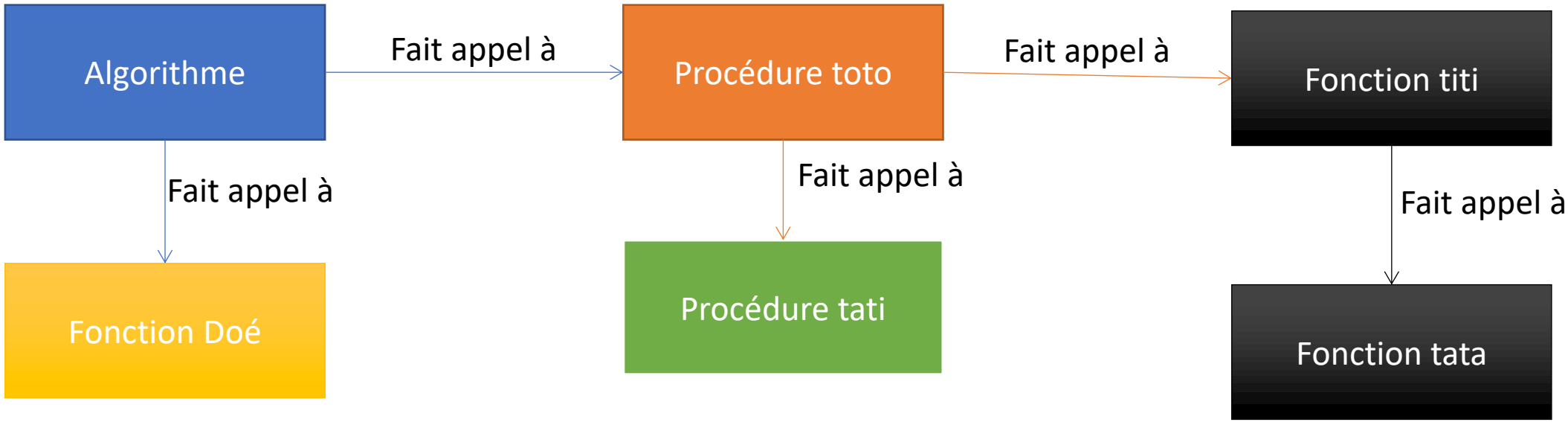


Une fonction comme une procédure sont toutes deux des algorithmes, la notion d'algorithme est utilisée ici pour désigner le programme qui fait appel à la fonction et/ou procédure.

1. Définitions

- Les fonctions ou procédures peuvent avoir besoin de paramètres pour s'exécuter tout dépend de leurs missions:
  - Par exemple une fonction qui doit calculer le carré d'un nombre a besoin, en paramètre du nombre dont elle doit calculer le carré, par contre une fonction qui doit demander à l'utilisateur son nom n'a pas besoin de paramètre lors de son appel.
  - Une procédure dont la mission est d'afficher le message « Hello nom_de_utilisateur » a besoin en paramètre qu'on lui fournisse le nom de l'utilisateur. Par contre une procédure qui doit afficher le message « Hello » n'a pas besoin de paramètre lors de son appel.

- Il peut arriver qu'une procédure ou fonction fasse appel à une autre fonction ou procédure lors de son exécution.



1. Définitions

- La notion de fonction et procédure est rencontrée couramment dans notre quotidien, quand un algorithme fait appel à une procédure ou fonction elle lui délègue une partie de ses missions. L'algorithme pour être fiable et utile doit savoir à qui il fait appel pour être sûr d'accomplir correctement sa mission de base.

- Imaginons que l'on commande un article sur un site en ligne. Qu'est ce qui se passe ? (exemple de transaction)
  - Le site nous fait la promesse de nous livrer l'article après paiement.
  - Il fait appel à un marchand qui possède l'article et achète avec lui l'article qu'il nous a promis.
  - Le marchand quant à lui fera appel à une compagnie de transport qui va se charger de nous livrer l'article.
  - Le client reçoit l'article très rapidement et sans problème. Ce client dira que le site est fiable sans pour autant chercher à savoir comment il a eu un tel résultat.

- Imaginons l'exemple précédent mais cette fois-ci la transaction se déroule mal (l'article est perdu lors du transport etc...) Qui est responsable ?
  - De la vue du client le responsable c'est le site.
  - De la vue du site le responsable est la compagnie de transport ou le vendeur.

- La différence avec les algorithmes, est qu'ici les transactions se dérouleront toujours bien sauf dans le cas ou le développeur fait appel à la mauvaise fonction ou écrit un algorithme qui ne fait pas ce qu'il doit faire.

Une erreur d'algorithme provient toujours d'une erreur humaine.

2. Notation des fonctions

- Pour définir une fonction en pseudo-code nous aurons besoin de définir :
  - Le nom de la fonction qui sera son étiquette.
  - Les paramètres que la fonction prend si nécessaire.
  - Le type de retour de la fonction.

- A la fin de chaque fonction on utilisera la notation **RENVOYER** suivi de la valeur de retour de la fonction après son traitement.

```
Fonction nom_de_la_fonction(parametre1 en Type_du_parametre, parametre2 en Type_du_parametre, ...) Retourne un type_de_retour
    Renvoyer valeur_de_retour
FinFonction

                Ou

FONCTION nom_de_la_fonction(parametre1 en Type_du_parametre, ...) RETOURNE un type_de_retour
    RENVOYER valeur_de_retour
FINFONCTION
```

Cette notation est inspirée de celle des langages typés, dans le cas des langages non typés le type de retour de la fonction n'est pas utile lors de la déclaration.

2. Notation des fonctions

- Lorsqu'une fonction termine son traitement, celle-ci retourne une valeur lors de son appel. Ainsi lors de l'appel de la fonction il faut affecter à la fonction une variable pour récupérer sa valeur de retour.

```
Début
    Variables nbre, double en Entiers
    nbre <- 2
    double <- CalculDouble(nbre)
    Ecrire "Le double de " + nbre + " est :" + double
Fin
```

- Ici on fait appel à une fonction **CalculDouble** qui prend un nombre entier en paramètre, calcule le double et retourne le résultat, la valeur de retour est stockée dans la variable **double** du programme appelant. **Quel est le type de la fonction lors de sa déclaration ?**

- Une fonction est un algorithme, on peut utiliser à l'intérieur de la fonction des conditions, des boucles, des déclarations .....

- Les variables déclarées à l'intérieur d'une fonction sont uniquement disponibles dans la fonction, pas en dehors. C'est comme dans le transport, un client qui achète un billet d'avion ne peut imposer ou savoir à l'avance le pilote de l'avion mais attend d'arriver à destination sans chercher à savoir l'organisation interne de la compagnie aérienne.

- La valeur de retour d'une fonction doit être identique au type de valeur qu'elle a déclarée renvoyer. De même lors de l'appel on doit veiller à ce que la valeur de retour de la fonction soit stockée dans une variable de type idéal.

3. Exemples utilisant les fonctions

Exemple 1: Ecrire une fonction qui prend en paramètre un nombre entier et qui retourne le double de ce nombre. Cette fonction sera appelée par un algorithme qui va recueillir la valeur auprès de l'utilisateur, appeler la fonction et afficher le résultat.

```
// Définition de la fonction
Fonction CalculDouble(nombre en Entier ) Retourne  un Entier
    Variable doubleDeNombre en Entier
    doubleDeNombre <- nombre * 2
    Renvoyer doubleDeNombre
FinFonction

// Définition de l'algorithme appelant
Début
    Variables nbre, double en Entiers
    Ecrire "Veuillez entrer un nombre:"
    Lire nbre
    double <- CalculDouble(nbre)
    Ecrire "Le double de " + nbre + " est " + double
Fin
```

3. Exemples utilisant les fonctions

Exemple 2: Ecrire une fonction qui doit demander à l'utilisateur son nom et renvoyer le nom. L'algorithme appelant doit afficher « Hello nom_utilisateur».

```
// Définition de la fonction
Fonction DemanderNom() Retourne une Chaîne
    Variable nom en Chaîne
    Ecrire "Quel est votre nom ?"
    Lire nom
    Renvoyer nom
FinFonction

// Définition de l'algorithme appelant
Début
    Variable nom en Chaîne
    nom <- DemanderNom()
    Ecrire "Hello " + nom
Fin
```

La variable « nom » de la fonction est différente de la variable nom de l'algorithme.

4. Notation des procédures

- Pour définir une procédure en pseudo-code nous aurons besoin de définir :
  - Le nom de la procédure qui sera son étiquette.
  - Les paramètres que la procédure prend si nécessaire.

```
Procédure nom_de_la_procédure(parametre1 en Type_du_parametre, parametre2 en Type_du_parametre, ...)
    ...

FinProcédure

                    Ou

PROCEDURE nom_de_la_procédure(parametre1 en Type_du_parametre, parametre2 en Type_du_parametre, ...)
    ...

FINPROCEDURE
```

Cette notation des procédures est uniquement dans le cadre de ce cours. Pour créer une procédure dans un langage de programmation, il faudra consulter la documentation du langage au préalable.

5. Exemple utilisant les procédures

Exemple 1: Ecrire un algorithme qui demande un nombre entier à l'utilisateur. L'algorithme fait appel à une procédure (méthode) qui va calculer le double du nombre et l'afficher.

```
// Définition de la procédure
Procédure calculEtAfficheDouble (nombre en Entier )
    Variable double en Entier
    double <- nombre * 2
    Ecrire "Le double de " + nombre + " est :" + double
FinProcédure

// Définition de l'algorithme appelant
Début
    Variable nombre en Entier
    Ecrire "Entrer un nombre:"
    Lire nombre
    calculEtAfficheDouble(nombre)
    Ecrire "Voilà , vous avez le résultat"
Fin
```

6. Exercices

Exercice 1

Ecrire un algorithme qui fait appel à une fonction dont le rôle est de demander à l'utilisateur de saisir un nombre positif. Le programme doit renvoyer le nombre positif saisi par l'utilisateur.

Attention : La fonction doit s'assurer que le nombre fourni est bien positif strictement supérieur à 0.

Exercice 2

Ecrire un algorithme qui fait appel à une procédure (méthode) dont le rôle est de demander le nom de l'utilisateur qui ne doit pas être vide c'est-à-dire différent de "" . Une fois la saisie du nom terminée, la méthode doit afficher « Hello nom_saisi_par_l'utilisateur ».

7. Corrections

Exercice 1

Ecrire un algorithme qui fait appel à une fonction dont le rôle est de demander à l'utilisateur de saisir un nombre positif. Le programme doit renvoyer le nombre positif saisi par l'utilisateur.

Attention : La fonction doit s'assurer que le nombre fourni est bien positif strictement supérieur à 0.

```
// Définition de la fonction
Fonction saisirNombre () Retourne un Entier
    Variable nombre en Entier
    Répéter
        Ecrire "Entrer un nombre positif:"
        Lire nombre
        Si nombre = 0 Ou nombre < 0 Alors
            Ecrire "Vous devez fournir un nombre positif"
        FinSi
    TantQue nombre = 0 Ou nombre < 0
    Renvoyer nombre
FinFonction


// Début de l'algorithme appelant
Début
    Variable nombreSaisi en Entier
    nombreSaisi <- saisirNombre()
    Ecrire "Vous avez saisi: " + nombreSaisi
Fin
```

7. Corrections

Exercice 2

Ecrire un algorithme qui fait appel à une procédure (méthode) dont le rôle est de demander le nom de l'utilisateur qui ne doit pas être vide c'est-à-dire différent de "" . Une fois la saisie du nom terminée, la méthode doit afficher « Hello nom_saisi_par_l'utilisateur ».

```
// Définition de la procédure
Procédure saisirNom()
    Variable nom en Chaîne
    nom <- ""
    Répéter
        Ecrire "Entrer votre nom:"
        Lire nom
        Si nom = "" Alors
            Ecrire "Vous devez fournir un nom."
        FinSi
    TantQue nom = ""
    Ecrire "Hello " + nom
FinProcédure

// Définition de l'algorithme appelant
Début
    saisirNom()
Fin
```

8. Projet : calcul de moyenne

Soit une classe de 10 élèves ayant chacun 5 notes en mathématiques. Le professeur aimerait avoir un algorithme qui lui permettra de :

- Saisir le nom des élèves
- Saisir les notes de chaque élève
- Calculer la moyenne de chaque élève

Le programme doit afficher pour chaque élève:

Elève 1: Nom = John ; Moyenne = 10,2

……

Elève 10: Nom = …; Moyenne =…

- **Le nom d'un étudiant ne doit jamais être vide. On considéra comme vide toute chaine égale à ""**
- **La note ne peut être négative ni supérieur à 20.**

9. Correction Projet

```
// Fonction qui demande le nom d'un élève
Fonction saisirNom(indice en Entier ) Retourne une Chaîne
    Variable nom en Chaîne
    nom <- ""
    Répéter
        Ecrire "Entrer le nom de l'élève " + indice  + ":"
        Lire nom
        Si nom = "" Alors
            Ecrire "Vous devez saisir un nom"
        FinSi
    TantQue nom = ""
    Renvoyer nom
FinFonction


// Fonction qui demande la note d'un élève
Fonction saisirNote(indice en Entier ) Retourne un Réel
    Variable note en Réel
    Ecrire "Entrer la note " + indice + ":"
    Lire note
    TantQue note < 0 Ou note > 20 Faire
        Ecrire "Vous devez saisir une note entre 0 et 20 svp."
        Ecrire "Entrer la note " + indice + ":"
        Lire note
    FinTantQue
    Renvoyer note
FinFonction
```

9. Correction Projet

```
// Fonction qui calcule la moyenne de notes
Fonction calculMoyenne(note1 en Réel ,note2 en Réel ,note3 en Réel ,note4 en Réel , note5 en Réel ) Retourne un Réel
    Variables moyenne, somme en Réels
    somme <- note1 + note2 + note3 + note4 + note5
    moyenne <- somme / 5
    Renvoyer moyenne
FinFonction

// Procédure qui affiche la description d'un élève
Procédure afficheDescription(indice en Entier ,nom en Chaîne ,moyenne en Réel )
    Ecrire "Elève " + indice + ": Nom = " + nom + "; Moyenne = " + moyenne
FinProcédure
```

- **Ces fonctions et procédures précédentes ainsi que l'algorithme final sont un exemple de résolution.**
- **Un sujet peut être résolu par différents types d'algorithmes.**

9. Correction Projet

```
// Ecriture de l'algorithme
Début
   Tableau tabNoms[10] en Chaînes
   Tableaux tabNotes[10][5], tabMoyenne[10] en Réels
   Variables compteur1, compteur2 en Entiers

   Ecrire "Bienvenue, sur l'algorithme de calcul de moyenne"

   // Commencer la saisie des noms
   Pour compteur1 <- 0 à 10 Pas 1 Faire
      tabNoms[compteur] <- saisirNom(compteur + 1)
   Suivant

   // Commencer la saisie des notes des élèves
   Pour compteur1 <- 0 à 10 Pas 1 Faire
      Ecrire "Saisie des notes de l'élève:" + tabNoms[compteur1]
      Pour compteur2 <- 0 à 5 Pas 1 Faire
         tabNotes[compteur1][compteur2] <- saisirNote(compteur2 + 1)
      Suivant
   Suivant

   // Commencer le calcul des moyennes
   Pour compteur1 <- 0 à 10 Pas 1 Faire
      tabMoyenne[compteur1] <- calculMoyenne(tabNotes[compteur1][0], tabNotes[compteur1][1], tabNotes[compteur1][2] ,tabNotes[compteur1][3], tabNotes[compteur1][4])
   Suivant

   // Afficher les résultats
   Pour compteur1 <- 0 à 10 Pas 1 Faire
      afficheDescription(compteur1 + 1, tabNoms[compteur1], tabMoyenne[compteur1])
   Suivant

   Ecrire "Merci d'avoir utilisé l'algorithme de calcul de moyenne"

Fin
```

Citations

« Le vrai problème n'est pas de savoir si les machines pensent, mais de savoir si les hommes pensent. » Skinner

« La plupart des gens trouvent le concept de la programmation évident, mais la réalisation impossible. » Anonyme

« La question de savoir si un ordinateur peut penser n'est pas plus intéressant que celle de savoir si un sous-marin peut nager. »Edgar W. Dijkstra