

L'algorithme

Pour créer un programme, après analyse de la problématique, il faut dans un premier temps le traduire en un algorithme.

L'algorithme est un moyen pour le programmeur de présenter son approche du problème à d'autres personnes.

En effet, c'est l'énoncé dans un langage bien défini d'une suite d'opérations permettant de répondre au problème. Il doit donc être :

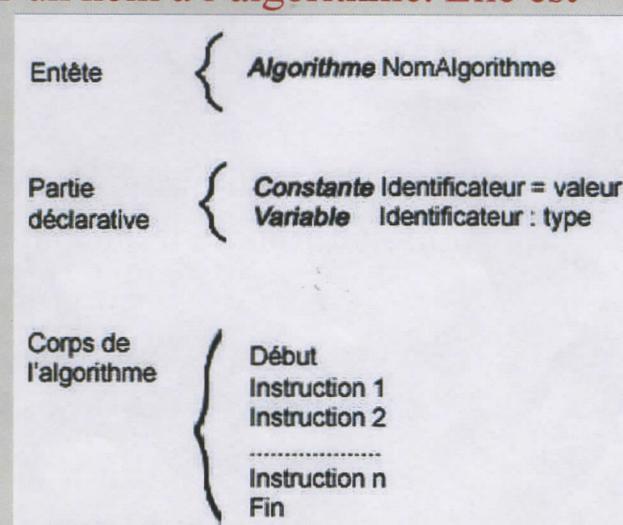
- Lisible : compréhensible même par un non-informaticien
- de haut niveau: être traduit en n'importe quel langage de programmation, chaque élément de l'algorithme ne doit pas porter à confusion
- concis: un algorithme ne doit pas dépasser une page. Sinon , il faut décomposer le problème en plusieurs sous-problèmes
- structuré: est composé de différentes parties facilement identifiables.

Le Cnam Rhône-Alpes - 4 rue Bayard - 69007 Lyon - 04 78 58 30 99

Structure générale d'un algorithme

Un **algorithme** est composé de trois parties principales :

- **L'en-tête** : cette partie sert à donner un nom à l'algorithme. Elle est précédée par le mot Algorithme ;
- **La partie déclarative** : dans cette partie, on déclare les différents objets que l'algorithme utilise (constantes, variables, etc.) ;
- **Le corps de l'algorithme** : cette partie contient les instructions de l'algorithme. Elle est délimitée par les mots Début et Fin.



Le Cnam Rhône-Alpes - 4 rue Bayard - 69007 Lyon - 04 78 58 30 99

La partie déclarative

Notion de variable

Les données et résultats des calculs sont rangés dans des cases mémoires qui correspondent à des variables.

Une variable est rangée dans un emplacement mémoire nommé, de taille fixe (ou non) prenant au cours de l'algorithme, un nombre indéfini de valeurs.

Déclaration des variables

Cette partie déclaration consiste à énumérer les variables dont on a besoin dans l'algorithme. Chaque déclaration comporte le nom de la variable (identificateur) et son type.

Syntaxe : **Variable identificateur : type**

Les différents types

Le type d'une variable est l'ensemble des valeurs qu'elle peut prendre.

Les différents types utilisés en algorithmique :

- **Type Entier** : sert à manipuler les nombres entiers positifs ou négatifs. **Type Réel** : sert à manipuler les nombres à virgule.
- **Type Caractère** : permet de manipuler des caractères alphabétiques et numériques. Exemple : 'a', 'A', 'z', '?', '1', '2', etc.
- **Type Chaîne** : sert à manipuler des chaînes de caractères. Exemple : "bonjour", "Monsieur", etc.
- **Type Logique (Booléen)** : utilise les expressions logiques. Il n'y a que deux valeurs booléennes : Vrai et Faux.

Exemple : Variables n : entier
 r : réel
 a,b : logiques
 nom_etudiant : chaîne

Opérations définies par type

Type	Opération possibles	Symbol ou mot correspondant
Entier	Addition Soustraction Multiplication Division Division entière Modulo (le reste de la division entière) x exposant y Comparaisons	+
		-
		*
		/
		(DIV en VB et % en C)
		(MOD en VB)
		$^$: en vb et pow(x,y) en C
		<, =, >, <=, >=, <>
En algorithmique nous symbolisons la division entière par DIV et le reste de la division entière par MOD		
Réel	Addition Soustraction Multiplication Division Exposant Comparaisons	+
		-
		*
		/
		$^$
		<, =, >, <=, >=, <>
Caractère	Comparaisons	<, =, >, <=, >=, <>
Chaîne	Concaténation Comparaisons	(+, & : en VB) <, =, >, <=, >=, <>
Booléen	Logiques	ET, OU, NON et OuEx

Le Cnam Rhône-Alpes - 4 rue Ravier - 69007 Lyon - 04 78 58 30 99

Les constantes

Comme une variable, à une constante correspond un emplacement mémoire réservé auquel on accède par le nom qui lui a été attribué, mais dont la valeur stockée ne sera jamais modifiée au cours du programme.

Syntaxe :

Constante NOM_DE_LA_CONSTANTE = valeur

Exemple :

Constante PI = 3.14

Les instructions de base

Une instruction est une action élémentaire commandant à la machine un calcul, ou une communication avec l'un de ses périphériques d'entrées ou de sorties. Les instructions de base sont :

- L'instruction d'affectation
- L'instruction d'entrée
- L'instruction de sortie
- Les commentaires

L'instruction d'affectation

L'affectation permet d'affecter une valeur à une variable. Elle est symbolisée en algorithmique par " \leftarrow ".

Le signe " \leftarrow " précise le sens de l'affectation.

Syntaxe :

Variable \leftarrow Expression

Exemple :

Algorithme	Calcul
Variables	A, B, C, D : entier
Début	
	A \leftarrow 10
	B \leftarrow 30
	C \leftarrow A+B
	D \leftarrow C*A
Fin	

L'instruction d'entrée

L'instruction d'entrée ou de lecture donne la main à l'utilisateur pour saisir une donnée. La valeur saisie sera affectée à une variable.

Syntaxe : **Saisir (identificateur)**

Exemples :

Saisir(A)

Saisir(A, B, C)

L'instruction Saisir(A) permet à l'utilisateur de saisir une valeur au clavier. Cette valeur sera affectée à la variable A.

L'instruction de sortie

Avant de lire une variable, il est conseillé d'écrire des libellés à l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper

L'instruction de sortie permet d'afficher des informations à l'écran.

Syntaxe : **Afficher(expression)**

Expression peut être une valeur, une résultat, un message, le contenu d'une variable, etc.

Exemple :

A ← 2

Afficher("La valeur de A est =",A)

La dernière instruction affiche à l'écran : La valeur de A est = 2

Exercice : Écrire un algorithme qui permet de saisir le prix HT d'un article et de calculer son prix total TTC, TVA = 20%.

Les commentaires

Lorsqu'un algorithme devient long, il est conseillé d'ajouter des lignes de commentaires dans l'algorithme, c'est à dire des lignes qui ont pour but de donner des indications sur les instructions effectuées et l'expliquer le fonctionnement du programme sans que le compilateur ne les prenne en compte.

Il y a trois types de commentaires, en général :

- // Commentaire sur une ligne
- /* Commentaire */
- ' Commentaire
- /* Commentaire
Sur plusieurs
lignes */

L'instruction conditionnel

L'instruction conditionnelle permet d'exécuter ou non une série d'instruction selon la valeur d'une condition.

Syntaxe : **Si** condition **Alors**
 Instruction(s) 1
Sinon
 Instruction(s) 2
FinSi

Une condition est une expression ou une variable logique évaluée à Vrai ou Faux.

Exercice 1 : Écrire un algorithme qui affiche si un nombre entier saisi au clavier est pair ou impair.

Exercice 2 : Écrire un algorithme qui demande deux nombres m et n à l'utilisateur et l'informe ensuite si le produit est négatif ou positif. On inclut dans l'algorithme le cas où le produit peut être nul.

L'instruction conditionnel

La structure Si ... Alors ... FinSi

Cette structure est utilisée si on veut exécuter une instruction seulement si une condition est vraie.

Syntaxe : Si Condition Alors
 Instruction(s)
 FinSi

Exercice :

Une grande surface accorde à ses clients, une réduction de 2% pour les montants d'achat supérieurs à 500,00 €.

Ecrire un algorithme permettant de saisir le prix total HT (PTHT) et de calculer le montant TTC (PTTC) en prenant en compte la remise et la TVA=20%.

L'instruction conditionnel

La structure Si ... Alors ... SinonSi ... Sinon FinSi

Syntaxe : Si condition1 Alors
 Instructions1
 SinonSi condition2 Alors
 Instructions2
 SinonSi condition3 Alors
 Instructions3 ...
 Sinon
 Instructions
 FinSi

L'instruction exécutée est celle dont la condition est vrai. Si aucune condition n'est vraie, c'est celle suivant le Sinon qui sera exécutée.

Exercice : Écrire un algorithme qui demande deux nombres m et n à l'utilisateur et l'informe ensuite si le produit est négatif ou positif. On inclut dans l'algorithme le cas où le produit peut être nul.

L'instruction conditionnel

Structure de choix multiples

Elle permet de choisir le traitement à effectuer en fonction de la valeur ou de l'intervalle de valeurs d'une variable ou d'une expression.

Syntaxe : **Suivant** sélecteur **faire**

 Valeur1 : action(s)1

 Valeur2 : action(s)2

...

 Valeurn : action(s)n

Sinon

 action(s)

FinSuivant

Lorsque l'ordinateur rencontre cette instruction, il vérifie la valeur de la variable de sélection et il la compare aux différentes valeurs.

Les valeurs sont évaluées dans l'ordre, et dès qu'une est vérifiée l'action associée est exécutée. L'instruction Sinon (facultative), sera exécutée si aucune des valeurs évaluées n'est remplie.

Exercice : Écrire algorithme qui affiche le mois en toute lettres selon son numéro saisi.

Les boucles itératives

La boucle itérative est utilisée quand une instruction ou une liste d'instructions, doit être répétée plusieurs fois. La répétition est soumise à une condition.

La boucle TantQue ... Faire

Elle permet de répéter un traitement tant que la condition est vraie.

Syntaxe : **TantQue** condition **Faire**

 Instruction(s)

FinTantQue

L'exécution de la boucle dépend de la valeur de la condition. Si elle est vraie, le programme exécute les instructions qui suivent, jusqu'à ce qu'il rencontre la ligne FinTantQue. Il retourne ensuite sur la ligne du TantQue, procède au même examen, et ainsi de suite.

La boucle ne s'arrête que lorsque la condition prend la valeur fausse, dans ce cas le programme poursuit son exécution après FinTantQue.

Exercice : Ecrire un algorithme qui calcule $S = 1 + 2 + 3 + 4 + 5 + \dots + N$.

Les boucles itératives

La boucle Pour ... Faire

La boucle Pour ... Faire permet de répéter une liste d'instructions un nombre connu de fois.

Syntaxe : **Pour** compteur **Allant de** v_initiale **à** v_finale (**Pas** v_pas) **Faire**
Instruction(s)

FinPour

La variable compteur est de type entier. Elle est initialisée à la valeur initiale. Le compteur augmente par défaut sa valeur de 1 à chaque tour de boucle jusqu'à la valeur finale, sauf si on précise une valeur de pas différentes (optionnel).

Pour chaque valeur prise par le compteur, la liste des instructions est exécutée.

Lorsque la variable compteur vaut la valeur finale, le traitement est exécuté une dernière fois puis le programme sort de la boucle.

Exercice : Écrire un algorithme qui saisit un nombre entier et qui calcule la somme des entiers pairs jusqu'à ce nombre.

Les boucles itératives

La boucle Répétez ... Jusqu'à

Cette boucle sert à répéter une instruction jusqu'à ce qu'une condition soit vraie.

Syntaxe : **Répéter**

Instruction(s)

Jusqu'à condition

La liste d'instructions est exécutées, puis la condition est évaluée. Si elle est fausse, le corps de la boucle est exécuté à nouveau puis la condition est réévaluée et si elle a la valeur vrai, le programme sort de la boucle et exécute l'instruction qui suit Jusqu'à.

Exercice : Ecrire un algorithme qui calcule la somme des N premiers nombres entiers. On suppose que N est strictement positif.

Les tableaux

Définition

Un tableau est une suite d'éléments de même type. Il utilise plusieurs cases mémoire à l'aide d'un seul nom. Comme toutes les cases portent le même nom, elles se différencient par un numéro ou un indice.

Nous pouvons représenter schématiquement un tableau nommé Note composé de cinq cases, dans la mémoire comme suit :

	Note[1]	Note[2]	Note[3]	Note[4]	Note[5]	Note[6]
Note	20	45	12,5	12	9	62

Contenu du tableau

Nous disposons alors de six variables. Pour les nommer, on indique le nom du tableau suivi de son indice entre crochets ou entre parenthèses : La première s'appelle Note[1], la deuxième Note[2], etc. jusqu'à la dernière Note[6].

Note[4] représente le 4ème élément du tableau Note et vaut 12

Tableau à une dimension

La déclaration d'un tableau permet d'associer à un nom une zone mémoire composée d'un certain nombres de cases mémoires de même type.

Syntaxe : Variable identificateur : tableau[indice_min .. indice_max] de type

Ou bien

Variable identificateur : tableau[taille] de type

- Le premier élément d'un tableau porte l'indice zéro ou 1 selon les langages
- La valeur d'un indice doit être un nombre entier.
- La valeur d'un indice doit être inférieure ou égale au nombre d'éléments du tableau. Exemple, avec le tableau tab[1 .. 20], écrire tab[0] et tab[21] est impossible.

Parcours complet d'un tableau

Les éléments d'un tableau étant indicés, on peut parcourir ces éléments avec une boucle, on utilisant une variable indice qui s'incrémenté à chaque tour de boucle.

Exercice : Écrire un algorithme permettant de saisir 30 notes et de les afficher après avoir multiplié toutes ces notes par un coefficient fourni par l'utilisateur.

Tableau à deux dimensions

Les tableaux à 2 dimensions se représentent comme une matrice ayant un certain nombre de lignes et un certain nombre de colonnes.

Nous pouvons représenter schématiquement un tableau de 3 lignes et de 4 colonnes comme suit :

Indices du Tableau	1	2	3	4
1	12	13	9	16
2	12.5	14	12	11
3	15	12	10	13

Syntaxe :

Variable identificateur : tableau[1..nb_lignes, 1..nb_colonnes] de type
ou bien

Variable identificateur : tableau[nb_lignes,nb_colonnes] de type

Exemple : L'instruction suivante déclare un tableau Note de type réel à deux dimensions composé de 3 lignes et de 4 colonnes :

Variables Note : tableau[1..3, 1..4] de réels

Tableau à deux dimensions

Pour accéder à un élément du tableau à deux dimensions, il suffit de préciser, entre crochets, les indices de la case contenant cet élément.

Exemple : L'instruction suivante affecte à la variable X la valeur du premier élément du tableau Note : $X \leftarrow \text{Note}[1, 1]$

Parcours complet d'un tableau à deux dimensions :

Pour parcourir une matrice nous avons besoin de deux boucles, l'une au sein de l'autre. La première boucle parcourt les lignes tandis que la deuxième parcourt les éléments de la ligne précisée par la boucle principale (la première boucle).

Exercice :

Écrire un algorithme permettant la saisie des notes d'une classe de 30 étudiants en 5 matières.

Tableau dynamique

Les tableaux dynamiques sont des tableaux dont la taille n'est définie que lors de l'exécution. Pour créer un tableau dynamique, il suffit de lui affecter une taille vide.

Syntaxe : Variable identificateur : tableau[] de type

Comme un tableau dynamique ne possède pas de taille prédéfinie, il convient de redimensionner le tableau avant de pouvoir s'en servir.

Syntaxe : Redimensionner identificateur[N]

Exemple :

Variable t : tableau[] d'entiers

Redimensionner t[11]

La première instruction déclare un tableau dynamique. La deuxième redimensionne la taille du tableau t à 11 éléments.

Les structures

Les structures permettent, contrairement aux tableaux, de désigner sous un seul nom un ensemble de valeurs pouvant être de types différents. L'accès à chaque élément de la structure nommé champ se fera par son nom au sein de la structure.

Déclaration d'une structure

Elle permet de définir un modèle de structure. Déclarer une structure c'est définir un nouveau type.

La déclaration des structures se fait dans une section spéciale des algorithme appelée Type, qui précède la section des variables.

Syntaxe :

```
Type Structure nom_structure
    nom_champ 1 : type_champ1
    ...
    nom_champN : type_champN
FinStructure
```

Les structures

Exemple : Déclaration d'une structure nommée étudiants

Type Structure étudiants

```
Nom : Chaîne
Prénom : Chaîne
Age : entier
Moyenne : réel
FinStructure
```

Déclaration d'une variable de type structure

Après avoir défini la structure, on peut l'utiliser comme un type normal en déclarant une ou plusieurs variables de ce type.

Syntaxe :

```
Variable nom_variable : nom_structure
```

Les structures

Exemple : On déclare deux variables Etud1 et Etud2 de type étudiants :

Variables Etud1, Etud2 : étudiants

Représentation : Les enregistrements sont composés de plusieurs zones de données, correspondant aux champs :

Etud1				
	Nom	Prénom	Age	Moyenne
Etud2				
	Nom	Prénom	Age	Moyenne

L'accès à un champ d'une structure

L'accès à un champ se fait en faisant suivre le nom de la variable de type structure du nom de champ séparé par un point.

Syntaxe : Nom_var.nom_champ

Exemple : Nous donnons un nom à l'étudiant 1, de la manière suivante :

Etud1.Nom ← "Abdel"

Exercice : Écrire un algorithme permettant de remplir la fiche de tous les étudiants de la classe.

Les fonctions prédéfinies

Tous les langages de programmation ont un certain nombre de fonctions qui permettent de connaître des résultats complexes.

Tout langage de programmation dispose d'un ensemble de fonctions prédéfinies permettant de procéder à des conversion de type de données, de calcul mathématiques, de manipulation de date et d'heures, de manipulation de chaînes de caractères et bien d'autres fonctions utiles.

Les fonctions de chaînes de caractères

- Longueur : **longueur(ch)** renvoie le nombre de caractères d'une chaîne
- Concaténation : **concat(ch1, ch2)** renvoie la concaténation de ch1 et de ch2
- Copie : **Copie(ch1, position, n)** recopie une partie de ch1 à partir de la position précisée, en limitant la recopie au nombre de caractères précisés par l'entier n.
- Comparaison : **comp(ch1, ch2)** compare deux chaînes de caractères
- Recherche: **recherche(ch1, ch2)** recherche l'occurrence d'un caractère ou une chaîne dans une chaîne de caractères

Les fonctions prédéfinies

Les fonctions mathématiques

Fonction	Description	Exemple	Résultat
Abs(nombre)	Retourne la valeur absolue d'un nombre	Abs(-12)	x = 12
Ent(nombre)	Retourne la partie entière d'un nombre	Ent(12.3)	x = 12
Cos(angle)	Retourne une valeur spécifiant le cosinus d'un angle	Cos(0)	x = 1
Sin(angle)	Retourne une valeur spécifiant le sinus d'un angle	Sin(0)	x = 0
Tan(angle)	Retourne une valeur contenant la tangente d'un angle	Tan(0)	x = 0
Sqrt(nombre)	Retourne une valeur spécifiant la racine carrée d'un nombre	Sqrt(4)	x = 2
Alea()	Retourne un nombre aléatoire compris entre 0 (inclus) et 1 (exclu)	alea()	0 =< x < 1

Procédures et fonctions

Lorsque l'algorithme devient complexe, on découpe l'algorithme en plusieurs parties plus petites. Ces parties sont appelées des sous-algorithmes.

Le sous-algorithme est écrit séparément du corps de l'algorithme principal et sera appellé par celui-ci quand ceci sera nécessaire. Il existe deux sortes de sous-algorithmes : les procédures et les fonctions..

Les procédures

C'est une série d'instructions regroupés sous un nom, qui permet d'effectuer des actions par un simple appel de la procédure dans un algorithme ou sous-algorithme.

Déclaration d'une procédure

Syntaxe : **Procédure nom_proc(liste de paramètres)**

Variables identificateurs : type

Début

Instruction(s)

FinRoc

Procédures et fonctions

Après le nom de la procédure, il faut donner la liste des paramètres avec leur type respectif. Ces paramètres sont appelés paramètres formels. Leur valeur n'est pas connue lors de la création de la procédure.

L'appel d'une procédure

Pour déclencher l'exécution d'une procédure dans un programme, il suffit de l'appeler. A l'appel d'une procédure, le programme interrompt son déroulement normal, exécute les instructions de la procédure, puis retourne au programme appelant et exécute l'instruction suivante.

Syntaxe : Nom_proc(liste de paramètres)

Les paramètres utilisées lors de l'appel d'une procédure sont appelés paramètres effectifs. Ces paramètres donneront leurs valeurs aux paramètres formels.

Exercice : Ecrire un algorithme permettant de dessiner un carré d'étoiles de 15 lignes et de 15 colonnes en faisant appel à une procédure qui affiche à l'écran une ligne de 15 étoiles puis passe à la ligne suivante.

Procédures et fonctions

Passage de paramètres

Les échanges d'informations entre une procédures et le sous algorithme appelant se font par l'intermédiaire de paramètres.

Il existe deux types de passages de paramètres qui permettent des usages différents :

Passage par valeur :

Le paramètre formel reçoit uniquement une copie de la valeur du paramètre effectif. La valeur du paramètre effectifs ne sera jamais modifiée.

Exemple : Soit l'algorithme suivant :

```
Algorithme Passage_par_valeur
Variables N : entier
//Déclaration de la procédure P1
Procédure P1(A : entier)
Début
    A ← A * 2
    Afficher(A)
FinProc
```

//Algorithme Principal
Début
 N ← 5
 P1(N)
 Afficher(N)
Fin

Procédures et fonctions

Passage par référence ou par adresse :

La procédure utilise l'adresse du paramètre effectif. Lorsqu'on utilise l'adresse du paramètre, on accède directement à son contenu. La valeur de la variable effectif sera donc modifiée.

Les paramètres passés par adresse sont précédés du mot clé Var.

Exemple : Reprenons l'exemple précédent :

Algorithme Passage_par_référence

Variables N : entier

//Déclaration de la procédure P1

Procédure P1 (Var A : entier)

Début

A ← A * 2

Afficher(A)

FinProc

```
//Algorithme Principal
Début
    N ← 5
    P1(N)
    Afficher(N)
Fin
```

Procédures et fonctions

Les fonctions

Les fonctions sont des sous algorithmes admettant des paramètres et retournant un seul résultat (une seule valeur) de type simple Déclaration d'une fonction

Syntaxe :

Fonction nom_Fonct (liste de paramètres) : type

Variables identificateur : type

Début

Instruction(s)

Retourner Expression

Fin

La syntaxe est assez proche de celle d'une procédure à laquelle on ajoute un type qui représente le type de la valeur renvoyée par la fonction et une instruction Retourner Expression. Cette dernière instruction renvoie au programme appelant le résultat de l'expression placée à la suite du mot clé Retourner.

Procédures et fonctions

L'appel d'une fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivie des paramètres effectifs. C'est la même syntaxe qu'une procédure.

A la différence d'une procédure, la fonction retourne une valeur. L'appel d'une fonction pourra donc être utilisé dans une instruction qui utilise sa valeur.

Syntaxe Nom_Fonc(list de paramètres)

Exercice : Ecrire un algorithme appelant 2 nombres et affichant , grâce à une fonction le plus grand de deux nombres différents.

La portée des variables

La portée d'une variable désigne le domaine de visibilité de cette variable. Une variable peut être déclarée dans deux emplacements distincts.

Une variable déclarée dans la partie déclaration de l'algorithme principale est appelée variable globale. Elle est accessible de n'importe où dans l'algorithme, même depuis les procédures et les fonctions. Elle existe pendant toute la durée de vie du programme.

Une variable déclarée à l'intérieur d'une procédure (ou une fonction) est dite locale. Elle n'est accessible qu'à la procédure au sein de laquelle elle définie, les autres procédures n'y ont pas accès. La durée de vie d'une variable locale est limitée à la durée d'exécution de la procédure.

```
Algorithme Portée
Variables X, Y : Entier
Procédure P1()
Variables A : Entier
Début
...
FinProc
//Algorithme principal
Début
...
Fin
```

X et Y sont des variables globales visibles dans tout l'algorithme.

A est une variable locale visibles uniquement à l'intérieur de la procédure

Avantages des procédures et fonctions

- Les procédures ou fonctions permettant de ne pas répéter plusieurs fois une même séquence d'instructions au sein du programme (algorithme).
- La mise au point du programme est plus rapide en utilisant des procédures et des fonctions. En effet, elle peut être réalisée en dehors du contexte du programme.
- Une procédure peut être intégrée à un autre programme, ou elle pourra être rangée dans une bibliothèque d'outils ou encore utilisée par n'importe quel programme.

Exercices

Exercice 1 : Simulation d'une calculatrice

Écrire un algorithme qui permet de saisir deux variables réelles a et b et un opérateur simple : $+$, $-$, $*$, $/$ et afficher le résultat.

Exercice 2 : Mention

Écrire un algorithme qui lit la moyenne générale (MG) d'un étudiant et affiche la mention.

Exercice 3 : Maximum de dix nombres

Écrire un algorithme qui permet d'afficher le maximum parmi dix nombres saisis au clavier.

Exercices

Exercice 4 : Nombre de moyennes ≥ 10

En utilisant les tableaux, écrire un algorithme qui permet la saisie d'une liste de n moyennes réelles et d'afficher le nombre des moyennes supérieures ou égales à 10. On suppose que $n \leq 100$.

Exercice 5 : Tableau à deux dimensions

Ecrire un algorithme qui permet :

La saisie des notes d'une classe de 5 étudiants en 4 matières

Calcul et affiche la moyenne de chaque étudiant

Calcul et affiche la moyenne de la classe dans chaque matière

Calcul et affiche la moyenne générale de la classe..

Exercice 6 : Structure imbriquée

Etudiant est une structure composée de trois champs : nom, prénom et date_de_naissance. Nom et prénom sont de type chaîne. Date_de_naissance est de type N_date. N_date est une structure composée de trois champs : jour, mois et année.

Jour et année de type entiers, mois est de type chaîne.

Ecrire un algorithme qui permet de saisir et d'afficher l'année de naissance d'un étudiant.

Exercices

Exercice 7 : Nombre de mots dans une phrase

Ecrire un algorithme permettant de saisir une chaîne de caractère (une phrase) et d'afficher le nombre de mots de cette chaîne.

Exercice 8 : Remplacement d'un caractère par un autre dans une phrase

Soit une phrase qui contient des mots séparés par des slashes « / », voir l'exemple suivant :

“Saunier / Remi / étudiant en troisième année / Tél : 067 98 09 34”

Ecrire un algorithme qui remplace les slashes « / » par des point virgules « ; ».