UNIVERSITE ASSANE SECK ZIGUINCHOR

Algorithmique et Programmation

LICENCE EN INGENIERIE INFORMATIQUE
2020 – 2021

Dr Ousmane DIALLO
odiallo@univ-zig.sn

PLAN

I. Introduction générale
II. Notions d'algorithme et de programme
III. Concepts de base de l'algorithmique
IV. Structures de contrôle
V. Les tableaux et les chaînes de caractères
VI. Les sous programmes

UNIVERSITE ASSANE SECK ZIGUINCHOR

Introduction générale

LICENCE EN INGENIERIE INFORMATIQUE
2020 – 2021

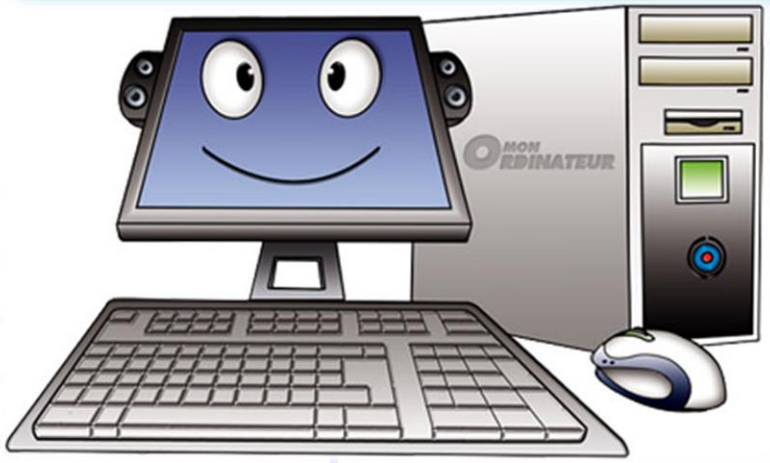
Dr Ousmane DIALLO
odiallo@univ-zig.sn

I. Introduction (1/17)

L'ordinateur est-il intelligent ?

I. Introduction (2/17)

L'ordinateur est-il intelligent ?




```
Program
Var nbre
        S, P : integer ;
        M : real ;
Begin
        write('donnez deux entiers :') ;
        read(nbre1, nbre2) ;
        S := nbre1 + nbre2 ;
```

```
Programme  calcul ;
Variable nbre1, nbre2: entier
        S, P : entier ;
        M : réel ;
Début
        écrire('donnez deux
entiers :') ;
        lire(nbre1, nbre2) ;
        S := nbre1 + nbre2 ;
        M := S / 2 ;
        P := nbre1 * nbre2 ;
        écrire(S, M, P) ;
```

1. Besoin d'automatisation

Utilisateur

4. Besoin automatisé

Programme

3. Automatiser le besoin : Une suite finie d'instructions pour satisfaire le besoin

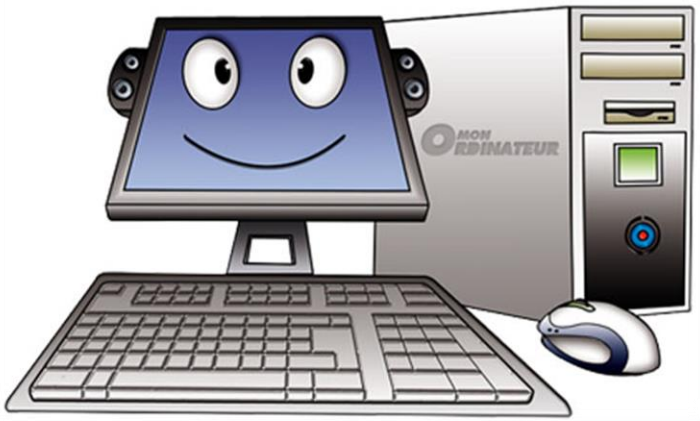
2. Étudier et Comprendre

Algorithme

Programmeur

I. Introduction (3/17)

L'ordinateur est-il intelligent ?



Pour quel point de vue ?

Non. Pour le programmeur

L'ordinateur ne sait rien faire à part ce qu'on lui a appris à faire.

Oui. Pour l'utilisateur

I. Introduction (4/17)

Utilisateur

- De manière générale, le mot **utilisateur** désigne une personne qui utilise quelque chose.
- En informatique, le terme **utilisateur** est employé pour désigner une personne qui utilise un système informatique.

I. Introduction (5/17)

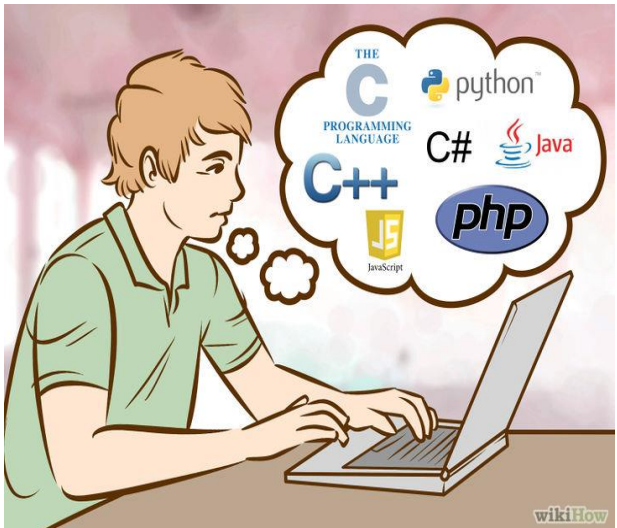
Besoin

- Le **besoin** recouvre l'ensemble de tout ce qui apparaît « *être nécessaire* » à un être.
- …

I. Introduction (6/17)

Programmeur

- Le programmeur (développeur) informatique est un professionnel des langages informatiques.
- Technicien ou ingénieur, il analyse les besoins des entreprises (utilisateurs) et crée des programmes sur mesure, ou améliore ceux qui existent.
- Une fois le logiciel créé, il le teste, en réalise la documentation technique, s'occupe du suivi et de la formation des utilisateurs.



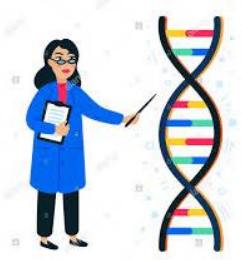
I. Introduction (7/17)

Algorithme et utilité

- Un ***algorithme*** est une suite d'actions ordonnées en séquence qui portent sur des objets bien définis et qui permettent de résoudre des classes de problèmes donnés.

Exemple:

- Le projet du génome humain pour identifier les 100 000 gènes de l'ADN humain, de déterminer les séquences des 3 milliards de paires de base chimiques….



- Internet permet à des gens éparpillés un peu partout dans le monde d'accéder rapidement à toutes sortes de données

I. Introduction (8/17)

Programme

- Un ***programme*** c'est le codage d'un algorithme dans un langage de programmation.
- C'est le programme qui est traité par une machine pour satisfaire ce besoin des utilisateurs.

I. Introduction (9/17)

Programmation

- La **programmation** informatique est l'ensemble des activités qui permettent **l'écriture des programmes informatiques** ou **le *codage* d'un logiciel.**
- On utilise aussi le terme **développement** pour dénoter l'ensemble des activités liées à la création d'un logiciel.

I. Introduction (10/17)

Ordinateur

I. Introduction (11/17)

Ordinateur

- Un **ordinateur** est une machine électronique programmable.
- Il fonctionne par la lecture séquentielle d'un ensemble d'instructions (organisées en programmes).
- Dès sa mise sous tension, un ordinateur exécute, l'une après l'autre, des instructions qui lui font lire, manipuler, puis réécrire un ensemble de données sur l'écran, des fichiers ou des bases de données.

I. Introduction (12/17)

Cas pratique

- La résolution de l'équation du second degré à solutions réelles $ax^2 + bx + c = 0$ ($a \neq 0$)
- Rendre automatique la résolution d'un problème revient à confier cette tâche à un ordinateur.
- En d'autres termes, il nous faut lui donner l'ensemble des instructions qu'il doit exécuter pour résoudre ce problème.
- C'est cet ensemble d'instructions que nous appellerons **programme**.

I. Introduction (13/17)

Cas pratique

- Et ce programme doit être écrit dans un langage compréhensible par l'ordinateur, par exemple le langage ***Pascal***.
- Mais, avant l'écriture d'un tel programme, nous devons d'abord élaborer une liste ordonnée des opérations qui, appliquées aux données du problème dans l'ordre prescrit, doivent aboutir à sa résolution.
- Cette suite d'opérations s'appelle **algorithme** et c'est la théorie des algorithmes que l'on nomme **algorithmique.**

I. Introduction (14/17)

Objectif Général

- Vous permettre d'appréhender les notions d'algorithme et de programmation.
- **THÈME : « PROGRAMMATION EN LANGAGE PASCAL AVEC TURBO-PASCAL OU DEV-PASCAL »**

I. Introduction (15/17)

Objectifs Spécifiques

- **À l'issu de ce cours, vous serez capable:**
  - De définir et d'expliquer les notions d'algorithme et de programmation ;
  - D'écrire des algorithmes et des programmes en langage PASCAL ;
  - D'utiliser un éditeur comme TURBO-PASCAL ou *DEV-PASCAL* pour éditer, compiler et exécuter un programme en PASCAL ;
  - D'utiliser les structures de contrôle, de déclarer et de manipuler les tableaux et de décomposer un programme en sous programmes.

I. Introduction (16/17)

Pré-requis

- Pour suivre ce cours vous devez avoir :
  - …

I. Introduction (17/17)

Stratégie Pédagogique

- Alternance de **cours théoriques** sous forme d'exposé avec des **Travaux dirigés** (Échanges avec les étudiants) et des **Travaux pratiques**.

UNIVERSITE ASSANE SECK ZIGUINCHOR

Fin
Introduction générale

LICENCE EN INGENIERIE INFORMATIQUE
2020 – 2021

Dr Ousmane DIALLO
odiallo@univ-zig.sn

UNIVERSITE ASSANE SECK ZIGUINCHOR

Chapitre 2
Notion d'algorithme et de programmation

LICENCE EN INGENIERIE INFORMATIQUE

2020 – 2021

Dr Ousmane DIALLO
odiallo@univ-zig.sn

PLAN

- Notion d'algorithme
- Propriétés d'un algorithme
- Expression des algorithmes
- Programmation informatique
- Langage de programmation
- Structures de données
- Structure générale d'un algorithme

II. Notions d'algo. Et progr.(1/18)

II.1. Algorithme

- L'***algorithmique*** est une science très ancienne. Son nom vient d'un mathématicien arabe du IXème siècle **Al-Khawarizmi**. ***L'algorithmique*** désigne actuellement la science qui étudie l'application des algorithmes à l'informatique.

Mais qu'est ce qu'un algorithme?

Définitions

- *Suite finie, séquentielle de règles que l'on applique à un nombre fini de données, permettant de résoudre des classes de problèmes semblables.*
  - *L'algorithme d'Euclide permet de trouver le P.G.C.D de deux nombre – Calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche.* ***(Petit Robert)***
- Il exprime les actions résolvant un problème donné **indépendamment des particularités de tel ou tel langage.**

II. Notions d'algo. Et progr.(2/18)

II.2. Expression des algorithmes (1/4)

- Un algorithme s'écrit le plus souvent dans un langage proche du langage courant, appelé **langage algorithmique** ou **pseudo-langage**.
- Cependant, il peut être écrit sous une forme schématique, appelée aussi **organigramme.**

II. Notions d'algo. Et progr.(3/18)

II.2. Expression des algorithmes (Pseudo-Code) (2/4)

- *Soit l'équation mathématique* ***" AX + B = 0 "****. Écrire un algorithme qui nous fait passer à une situation finale acceptable si notre situation initiale est acceptable.*

Analyse:

Description des situations initiales acceptables :

- A et B sont des valeurs réelles bien définies et contenues dans des variables de même nom.

Description des situations finales acceptables :

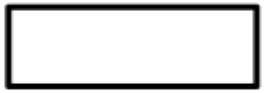
- A et B n'ont pas été modifiées.
- Nous avons affiché " Tout réel est solution. " si et seulement si tout réel est solution de l'équation (A=0 et B=0).
- Nous avons affiché " Pas de solution. " si et seulement si l'équation possède pas de solution (A=0 et B$\neq$0).
- Nous avons affiché " La solution de l'équation est –B/A. " si et seulement si l'équation possède effectivement une solution (A $\neq$0).
- Rien d'autre n'a été affiché ou imprimé depuis la situation initiale.

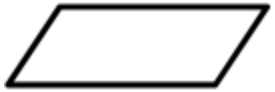
II. Notions d'algo. Et progr.(4/18)

II.2. Expression des algorithmes (organigramme) (3/4)

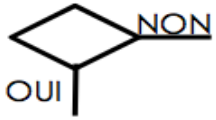
- La représentation schématique d'un algorithme est couramment désignée par le terme **organigramme** ou logigramme (norme **NF Z 67-010**).
- L'organigramme permet donc la mise en œuvre de symboles représentant des traitements, des données, des liaisons... Il présente l'intérêt d'une visualisation globale mais reste limité aux études peu complexes (s'il ne tient plus sur une page, l'organigramme devient illisible....). En voici quelques symboles:


début, fin, interruption

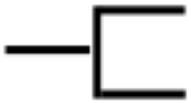

opérations, ou groupe d'opérations


entrées/sorties (lecture ou écriture)


Jonction



embranchement (test), conditions variables

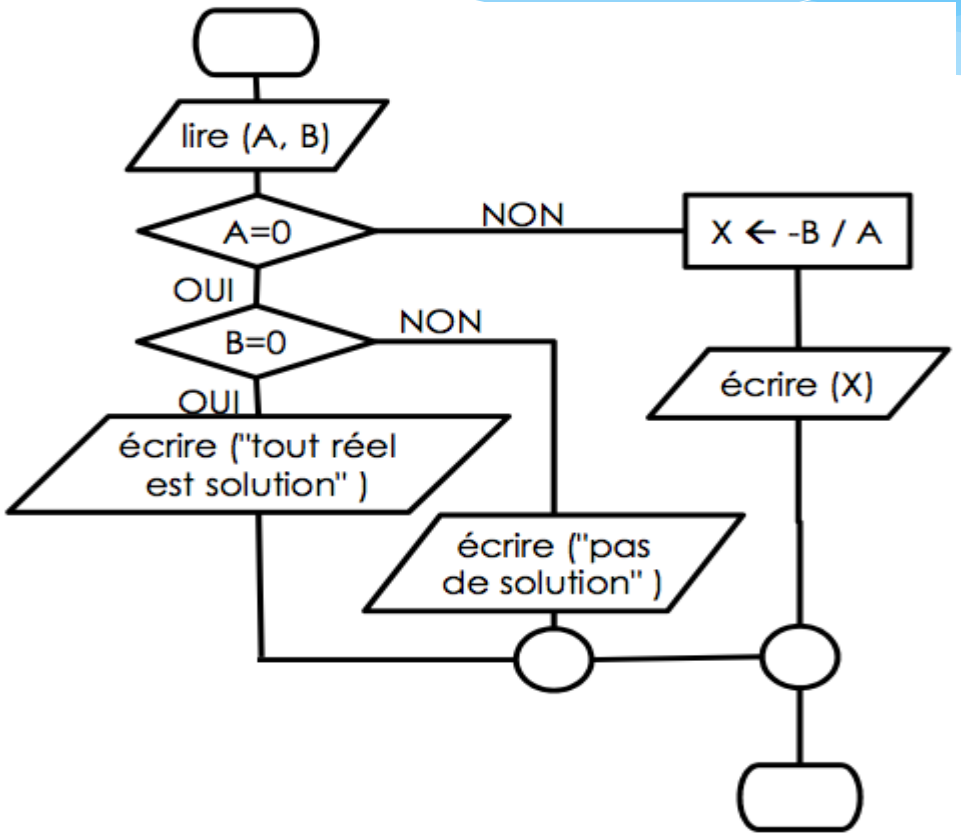

commentaires (indications)

Le sens général doit être de haut en bas, de gauche à droite (ex eq. 1er degré...).

II. Notions d'algo. Et progr.(5/18)

II.2. Expression des algorithmes (organigramme) (4/4)

- Organigramme de résolution de l'équation du premier degré ***AX + B = 0***.



II. Notions d'algo. Et progr.(6/18)

II.3. Propriétés d'un algorithme (1/2)

Un algorithme doit être :

- **PRECIS**: Il doit indiquer:
  - l'ordre des étapes qui le constituent
  - à quel moment il faut cesser une action
  - à quel moment il faut en commencer une autre
  - comment choisir entre différentes possibilités
- **DETERMINISTE**
  - Une suite d'exécutions à partir des mêmes données doit produire des résultats identiques.
- **FINI DANS LE TEMPS ET PRODUCTIF**
  - c'est-à-dire s'arrêter au bout d'un temps fini en fournissant le résultat escompté

II. Notions d'algo. Et progr.(7/18)

II.3. Propriétés d'un algorithme (2/2)

- Exemple : Algorithme de résolution de l'équation du premier degré A X + B = 0

```
Lire les coefficients A et B
Si A est non nul
        alors
           affecter à X la valeur - B / A
           afficher à l'écran la valeur de X
        sinon
          si B est nul
           alors
              écrire "tout réel est solution"
            sinon
               écrire "pas de solution"
```

Précis
Suite linéaire d'opérations bien définies

Déterministe
Avec les mêmes coefficients on obtiendra le même résultat

Fini et productif
Se termine et donne pour chaque cas un résultat

II. Notions d'algo. Et progr.(8/18)

II.4. La programmation informatique (1/4)

- Les différentes phases de l'analyse et de la programmation informatique (implémentation sous forme de programme informatique d'un algorithme ou d'un procédé bien formalisé):

1. **Analyse du problème**
2. **Conception d'une solution** : algorithmique, choix de la représentation des données, choix de la méthode utilisée,
3. **Développement :** programmation, choix du langage de programmation, choix de la machine utilisée
4. **Tests et validation**
5. **Maintenance**
6. **Documentation du programme**

II. Notions d'algo. Et progr.(9/18)

II.4. La programmation informatique (2/4)

- **Analyse du problème:**

  L'analyse consiste à bien comprendre l'énoncé du problème: Il est inutile et dangereux de passer à la phase suivante si vous n'avez pas bien discerné le problème. La question à se poser : **QU'EST-CE?**

- **Conception d'une solution:**

  C'est la phase algorithmique où l'on finalise les choix conceptuels, les structures et outils informatiques à utiliser pour résoudre le problème. Les questions : **AVEC QUOI? COMMENT?**

- **Le développement:**

  Plusieurs langages de programmation et architectures informatiques seront disponibles pour implémenter **l'algorithme qui ne doit être lié à aucun d'eux**. Il faut choisir ces éléments en restant cohérent avec la solution proposée.

II. Notions d'algo. Et progr.(10/18)

II.4. La programmation informatique (3/4)

- **Tests et validation:**

Vérifier l'exactitude du comportement de l'algorithme, son bon déroulement.

Si l'algorithme ne répond pas parfaitement à toutes les requêtes exprimées dans l'énoncé du problème, retournez à la phase n°1.

- **Maintenance:**

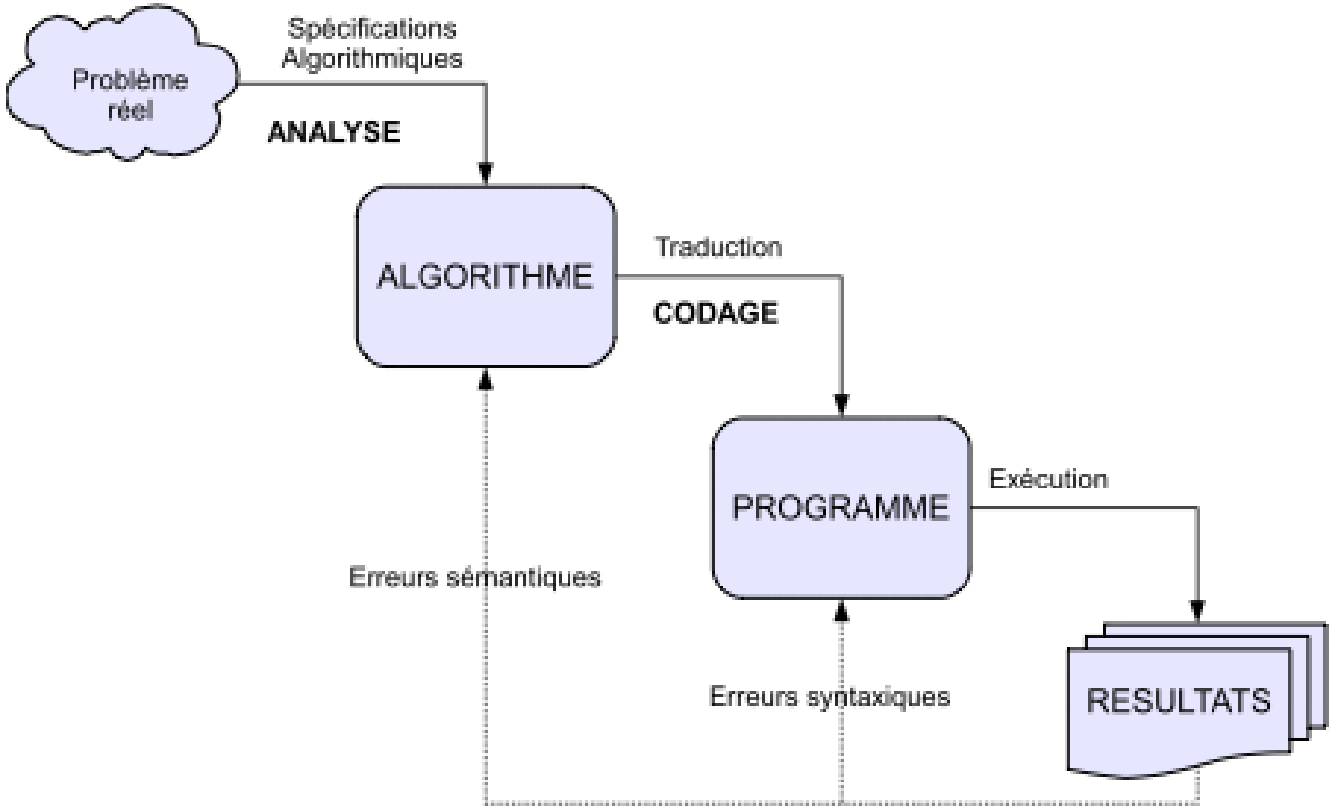
activité qui assure l'entretien et le bon fonctionnement du programme tant et aussi longtemps qu'il sera utilisé.

- **Documentation du programme:**

chaque étape énoncée doit être documentée pour assurer la pérennité du programme.

II. Notions d'algo. Et progr.(11/18)

II.4. La programmation informatique (4/4)



Les différentes étapes du processus de programmation

II. Notions d'algo. Et progr.(12/18)

II.5. Langage de programmation (1/4)

- Pour être compris et exécuté par un ordinateur, un algorithme doit être traduit dans un langage spécifique, qu'on appelle **langage de programmation**.

- On obtient ainsi ce qu'on appelle un **programme informatique** qui contient l'ensemble des actions consécutives que l'ordinateur doit exécuter. Ces actions sont appelées **instructions**.

II. Notions d'algo. Et progr.(13/18)

II.5. Langage de programmation (2/4)

- Un programme en langage machine est uniquement constitué d'une suite de 0 et de 1 (code en binaire).
- Il est plus pratique de réaliser un programme dans un langage plus compréhensible par l'homme et de le traduire ensuite en langage machine.
- Selon la méthode de traduction, on distingue:
  - Les ***langages compilés***
  - Les ***langages interprétés***.

II. Notions d'algo. Et progr.(14/18)

II.5. Langage de programmation (3/4)

Les langages compilés

- Dans le cas d'un langage compilé (exemples: C, C++, Pascal...) , le programme réalisé, appelé programme source, est traduit complètement par ce qu'on appelle un **compilateur** avant de pouvoir être exécuté. La compilation génère un programme dit **exécutable**.

- Ce programme généré est **autonome**, c'est-à-dire qu'il n'a pas besoin d'un autre programme pour s'exécuter. Mais à chaque modification du fichier source (le programme source) il faudra le recompiler pour que les modifications prennent effet.

II. Notions d'algo. Et progr.(15/18)

II.5. Langage de programmation (4/4)

Les langages interprétés

- Un programme écrit dans un langage interprété (exemples: Perl, Lisp, Prolog...) a besoin, pour chaque exécution, d'un programme annexe appelé **interpréteur** qui va lire le code source pour traduire et faire exécuter une à une chacune des instructions. **Dans ce cas, il n'y a pas de génération de programme exécutable.**

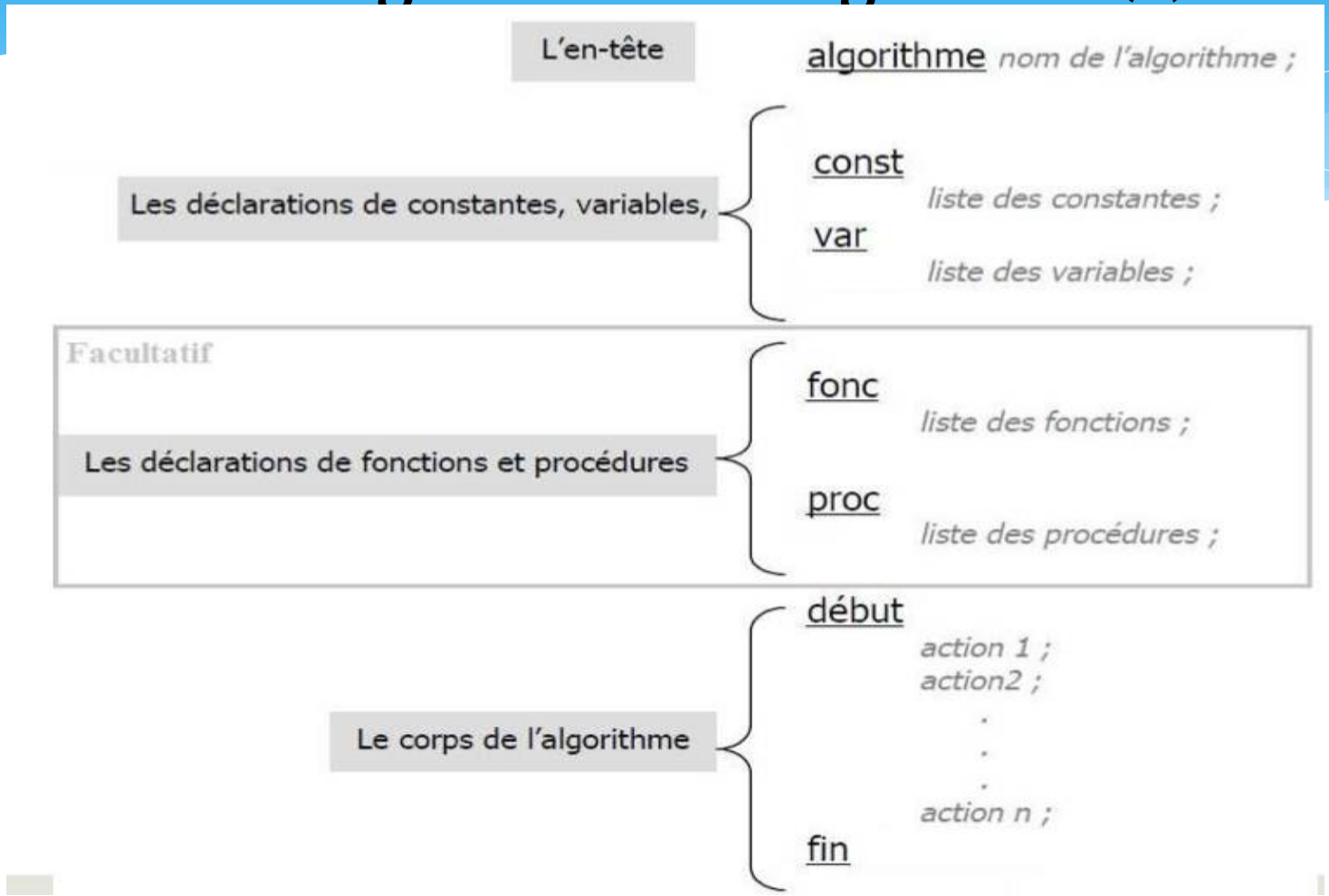
II. Notions d'algo. Et progr.(16/18)

II.6. Structures de données

- Une structure de données est une façon d'organiser des informations dans la mémoire de l'ordinateur pour en faciliter leur manipulation
- Il en existe plusieurs:
  - Tableau,
  - Structure
  - liste chaînée, etc.
- Il existe une forte dépendance entre structures de données et programmes:
  - influence sur la **simplicité** du programme
  - influence sur l'**efficacité** du programme

II. Notions d'algo. Et progr.(17/18)

II.7. La structure générale d'un algorithme (1/2)



II. Notions d'algo. Et progr.(18/18)

II.7. La structure générale d'un algorithme (2/2)

- **L'en-tête :** permet d'identifier tout simplement l'algorithme (donner un nom en un seul mot ou plusieurs mots composés reliés par – ou _)
- **Les déclarations :** liste exhaustive des objets (constantes et variables principalement), structures et grandeurs utilisés et manipulés dans le corps de l'algorithme. Elle doit être située en début de l'algorithme **(tout doit être déclaré avant d'être utilisé)**
- **Le corps :** contient les instructions et opérations à exécuter une à une.
- **Les commentaires:** éventuellement présents dans l'algorithme, ils permettent de détailler, de commenter l'algorithme. Ces lignes de commentaires placées entre **(* et *)** ne sont nullement exécutées, elles sont juste utilisées à titre d'information pour le programmeur (ou pour un autre utilisateur de l'algorithme).

FIN CHAP2

UNIVERSITE ASSANE SECK ZIGUINCHOR

Chapitre 3
Concepts de base de l'algorithmique

LICENCE EN INGENIERIE INFORMATIQUE
2020 – 2021

Dr Ousmane DIALLO
odiallo@univ-zig.sn

PLAN

- Notion d'objet, d'identificateur et de donnée
- Constantes et variables
- Types de données
- Opérateur, opérande et expression
- Actions sur les variables
- Les entrées-sorties
- Notre 1er algorithme
- Exemples d'application
- Corrections des exemples d'application

III. Concepts de base de l'algorithmique(1/34)

III.1. Notions d'objet, d'identificateur et de donnée (1/3)

Notion d'objet:

- Un objet est un élément manipulé dans un algorithme ou programme. Il peut s'agir de constantes, variables...
- Un objet est parfaitement défini, si nous connaissons ses trois caractéristiques, à savoir :
  - son **identificateur;**
  - Son **type;**
  - et sa **valeur**.

III. Concepts de base de l'algorithmique(2/34)

III.1. Notions d'objet, d'identificateur et de donnée (2/3)

Notion d'identificateur:

- Un identificateur est un nom donné à un élément du programme.
  - Nom d'une variable, constante, programme….
- Il est représenté par une suite quelconque de caractères alphanumériques sans espace.
  - En un seul mot ou plusieurs mots composés reliés par – ou _
- De préférence, le nom est choisi en rapport avec le contenu de l'objet.

III. Concepts de base de l'algorithmique(3/34)

III.1. Notions d'objet, d'identificateur et de donnée (3/3)

Notion de donnée:

- Une donnée est une **valeur** introduite par l'utilisateur pendant l'exécution du programme.
- Elle peut être lue interactivement ou issue d'un fichier de données.
  - La température, l'âge du patient,…

III. Concepts de base de l'algorithmique(4/34)

III.2. Les constantes et les variables (1/2)

Les constantes:

- Représentent des nombres, des chiffres, des caractères, des chaines de caractères. ***Une fois initialisées, leurs valeurs ne peuvent pas être modifiées*** tout au long de l'exécution de l'algorithme.

- Une constante est définies par:
  - **l'identificateur:** correspond au nom et est composé de lettres et de chiffres
  - **le type:** Une constante possède un type déterminé par sa valeur
  - **Une valeur:** la valeur de la constante

III. Concepts de base de l'algorithmique(5/34)

III.2. Les constantes et les variables (2/2)

Les variables:

- Les variables sont des entités manipulées par le programme lors de son exécution.
- Elles correspondent à des **emplacements** situés dans la mémoire vive de l'ordinateur et permettant de stocker des valeurs (données).
- Une variable est caractérisée par:
  - Son ***identificateur***: son nom,
  - Son ***type***: détermine la nature de l'information (nombre, caractère, etc.),
  - Sa ***valeur***: le contenu de la variable.

La valeur d'une variable peut être modifiée durant l'exécution du programme.

III. Concepts de base de l'algorithmique(6/34)

III.3. Les types de données (1/2)

- Le type d'une variable caractérise :
  - l'ensemble des valeurs que peut prendre la variable
  - l'ensemble des actions que l'on peut effectuer sur une variable
- Une fois qu'un type de données est associé à une variable le contenu de cette variable doit **obligatoirement être du même type**
- Il y a deux grandes catégories de type :
  - les **types simples**
  - les **types complexes** (que nous verrons dans la suite du cours)

III. Concepts de base de l'algorithmique(7/34)

III.3. Les types de données (2/2)

Les types simples (types de base)

- En algorithmique, on distingue cinq types de base:
  - **L'entier (mot clé : entier):** les nombres entiers positifs ou négatifs
  - **Le réel (mot clé : réel) :** les nombres réels positifs ou négatifs
  - **Le caractère (mot clé : char) :** lettre, chiffre, ponctuation, espace,… et plus généralement toutes les touches que l'on peut trouver sur une machine à écrire
  - **La chaine de caractères (mot clé : chaine):** Ce type se compose d'une suite de symboles de type caractère.
  - **Le type booléen (mot clé : booléen): VRAI, FAUX** : C'est un type logique qui peut prendre que les valeurs VRAI ou FAUX. Il servira pour les expressions et les conditions.

III. Concepts de base de l'algorithmique(8/34)

III.4. Déclaration des variables et des constantes (1/2)

- Toute variable ou constante doit être déclarée avant sa première utilisation.

Déclaration d'une constante

- La déclaration d'une constante est toujours associée à son initialisation.
- La syntaxe de déclaration d'une constante est:

constantes

identificateur : type <- valeur;

- **Exemple**

constante MAX : entier <- 32767;

III. Concepts de base de l'algorithmique(9/34)

III.4. Déclaration des variables et des constantes (2/2)

Déclaration d'une variable

- La **déclaration** d'une variable consiste à la réservation d'un emplacement mémoire, auquel on donne un nom unique appelé **identificateur** et par lequel on peut accéder à sa valeur.
- La syntaxe de déclaration d'une variable est:

variables identificateur : type;

- **Exemple**

variable a : entier;

Cette déclaration permet de réserver un emplacement mémoire pour le stockage d'un entier qui sera nommé **a** dans la suite du programme.

III. Concepts de base de l'algorithmique(10/34)

III.5. Structure générale d'un algorithme(rappel)

Programme *nom du programme*

Constante

Déclaration de constantes

Type

Définition de types

Variable

Déclaration de variables

Définition de sous programmes

Début

instructions

Fin

III. Concepts de base de l'algorithmique(11/34)

III.6. Opérateurs, opérandes et expressions (1/14)

- Un ***opérateur*** est un symbole d'opération qui permet d'agir sur des variables ou de faire des calculs.
- Un ***opérande*** est une entité (variable, constante ou expression) utilisée par un opérateur
- Par exemple dans ***a + b*** :
  - a est l'opérande gauche
  - + est l'opérateur
  - b est l'opérande droite

III. Concepts de base de l'algorithmique(12/34)

III.6. Opérateurs, opérandes et expressions (2/14)

Les opérations en algorithme

Type	Exemples de valeurs	opérations possibles	opérateur ou mot clé correspondant
réel	-15.69 , 0.36	addition, soustraction, multiplication, division, comparaison	+, -, *, /, <, ≤, >, ≥, = , ≠
entier	-10 , 0 , 3 , 689	addition, soustraction, multiplication, division, modulo, comparaison	+, -, *, div, mod, <, ≤, >, ≥, = , ≠
caractère	'B' , 'h' , '£' , '?'	successeur, prédécesseur, ordre, caractère, comparaison	suc, pred, ord, car, <, ≤, >, ≥, = , ≠
chaîne	"Bonjour" , "93000", "toto@caramail.com"	longueur, concaténation, comparaison	longueur , + , <, ≤, >, ≥, =, ≠
booléen	VRAI , FAUX	négation, conjonction , disjonction	NON , ET , OU

III. Concepts de base de l'algorithmique(13/34)

III.6. Opérateurs, opérandes et expressions (3/14)

Opérateurs

- Un opérateur peut être ***unaire*** ou ***binaire***:
  - ***Unaire*** s'il n'admet qu'un seul opérande, par exemple l'opérateur ***non***
  - ***Binaire*** s'il admet deux opérandes, par exemple l'opérateur ***+***
- Un opérateur est associé à ***un type de donnée*** *et ne peut être utilisé qu'avec* des variables, des constantes, ou des expressions de *ce type*
  - Par exemple l'opérateur + ne peut être utilisé qu'avec les types arithmétiques (naturel, entier et réel) ou (exclusif) le type chaîne de caractères dans le cas des concaténations.
  - **On ne peut pas additionner un entier et un caractère**

III. Concepts de base de l'algorithmique(14/34)

III.6. Opérateurs, opérandes et expressions (4/14)

Opérateurs

- Toutefois ***exceptionnellement*** *dans certains cas on accepte d'utiliser un opérateur* avec deux opérandes de types différents, c'est par exemple le cas avec les types arithmétiques (2+3.5)

❑ La signification d'un opérateur peut changer en fonction du type des opérandes

- Par exemple l'opérateur **+** avec des entiers aura pour sens l'addition, mais avec des chaînes de caractères aura pour sens la **concaténation**
  - 2+3 vaut 5
  - "bonjour" + " à tous" donne "bonjour à tous "

III. Concepts de base de l'algorithmique(15/34)

III.6. Opérateurs, opérandes et expressions (5/14)

Opérateurs sur les entiers et réels

- On retrouve tout naturellement +, -, /, *
- Avec en plus pour les entiers **div** et **mod**, qui permettent respectivement de calculer une *division entière* et le *reste de cette division*, par exemple :
  - ***11 div 2*** vaut 5
  - ***11 mod 2*** vaut 1

III. Concepts de base de l'algorithmique(16/34)

III.6. Opérateurs, opérandes et expressions (6/14)

Opérateurs sur les énumérés

❑ Pour les énumérés nous avons trois opérateurs **succ, pred, ord** :

- ***succ*** permet d'obtenir le successeur, par exemple avec le type JourDeLaSemaine :
  - ***succ*** Lundi vaut Mardi
  - ***succ*** Dimanche vaut Lundi
- ***pred*** permet d'obtenir le prédécesseur, par exemple avec le type JourDeLaSemaine :
  - ***pred*** Mardi vaut Lundi
  - ***pred*** Lundi vaut Dimanche
- ***ord*** permet d'obtenir le numéro d'ordre de l'énuméré spécifié, par exemple avec le type JourDeLaSemaine :
  - ***ord*** Lundi vaut 0
  - ***ord*** Dimanche vaut 6

III. Concepts de base de l'algorithmique(17/34)

III.6. Opérateurs, opérandes et expressions (7/14)

Opérateurs sur les caractères

- Pour les caractères on retrouve les trois opérateurs des énumérés avec en plus un quatrième opérateur nommé ***car*** qui est le dual de l'opérateur ***ord*** avec comme fonction de bijection la table de correspondance de la norme ASCII
  - Par exemple
    - ***ord*** A vaut 65
    - ***car*** 65 vaut A
    - ***pred*** A vaut @

III. Concepts de base de l'algorithmique(18/34)

III.6. Opérateurs, opérandes et expressions (8/14)

Opérateurs booléens

- Pour les booléens nous avons les operateurs ***non, et, ou, ou Exclusif***

Non

a	non a
Vrai	Faux
Faux	Vrai

ET

a	b	a et b
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

OU

a	b	a ou b
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

OU Exclusif

a	b	a ouExclusif b
Vrai	Vrai	Faux
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

III. Concepts de base de l'algorithmique(19/34)

III.6. Opérateurs, opérandes et expressions (9/14)

Opérateur d'égalité, d'inégalité, etc.

- L'opérateur d'égalité
  - C'est l'opérateur que l'on retrouve chez tous les types simples qui permet de savoir si les deux opérandes sont égaux
  - Cet opérateur est représenté par le caractère **=**
  - Une expression contenant cet opérateur est un booléen
- On a aussi l'opérateur d'inégalité **!=**
- Et pour les types possédant un ordre les opérateurs de comparaison <, >, <=, >=

III. Concepts de base de l'algorithmique(20/34)

III.6. Opérateurs, opérandes et expressions (10/14)

Notion d'expression

- Une expression est:
  - soit une variable ou une constante,
  - soit une valeur (par exemple : "bonjour", 45),
  - soit une combinaison de variables, de constantes, de valeurs et d'opérateurs (par exemple : 2 * r * 3.14)
- Elle est évaluée durant l'exécution du programme, et possède une valeur (son évaluation) et un type.
- Par exemple, supposons que *rayon* soit une variable de valeur 5. Alors l'expression 2 * *rayon* * 3.14 vaut 31.4

III. Concepts de base de l'algorithmique(21/34)

III.6. Opérateurs, opérandes et expressions (11/14)

Les expressions conditionnelles

- Une expression conditionnelle (ou expression logique, ou expression booléenne ou condition) est une expression dont la valeur est soit ***VRAI*** soit ***FAUX***.
- On peut donc affecter une expression conditionnelle à une variable booléenne. Il existe plusieurs types d'expressions conditionnelles:
  - **Les *conditions simples*:**
  - **Les *conditions complexes*:**

III. Concepts de base de l'algorithmique(22/34)

III.6. Opérateurs, opérandes et expressions (12/14)

Les expressions conditionnelles

- **Les *conditions simples*:** une condition simple est une comparaison de deux expressions de même type.
- **Exemples:**
  - a<0 , op = 's' , (a + b - 3) * c = (5 * y –2) / 3
- **Les *conditions complexes*:** formées de plusieurs conditions simples ou variables booléennes reliées entre elles par les opérateurs logiques **ET, OU, NON.**
- **Exemples:**
  - *(a<0)**ET**(b<0)*
    *((a +3 = b) **ET** (c < 0)) **OU** ((a = c *2) **ET** (b!= c))*

III. Concepts de base de l'algorithmique(23/34)

III.6. Opérateurs, opérandes et expressions (13/14)

Priorité des opérateurs

- Tout comme en arithmétique les opérateurs ont des priorités
  - Par exemple * et / sont prioritaires sur + et –
- Pour les booléens, la priorité des opérateurs est non, et, ou Exclusif et ou
- Pour clarifier les choses (ou pour dans certains cas supprimer toutes ambiguïtés) on peut utiliser des parenthèses

III. Concepts de base de l'algorithmique(24/34)

III.6. Opérateurs, opérandes et expressions (14/14)

Récapitulatif des opérateurs sur les types de base

Opérateur	Notation	Type des opérandes	Type du résultat
+ et – unaires négation logique	+ - NON	entier ou réel booléen	celui de l'opérande booléen
Elévation à la puissance	↑	entier ou réel	entier ou réel
Multiplication Division entière Division reste(modulo)	* DIV / MOD	entier ou réel entier entier ou réel entier	entier ou réel entier réel entier
Addition Soustraction	+ -	entier ou réel	entier ou réel
Comparaison	< <= > >= = <>	tout type	booléen
et logique ou logique	ET OU	booléen	booléen

III. Concepts de base de l'algorithmique(25/34)

III.7. Actions sur les variables (1/2)

- On ne peut faire que deux choses avec une variable :

1. Obtenir son contenu
   - Cela s'effectue simplement en nommant la variable
2. Affecter un (nouveau) contenu (mettre une nouvelle information dans la variable)
   - Cela s'effectue en utilisant l'opérateur d'affectation représenté par le symbole

III. Concepts de base de l'algorithmique(26/34)

III.7. Action sur les variables (2/2)

L'affectation

- Elle permet d'affecter une valeur à une variable. La valeur doit être compatible avec le type de la variable.

Identificateur ⟵ expression; ou identificateur := expression;

X ⟵ 1; (X prend la valeur 1)

X ⟵ 2*3+5; (X prend la valeur du résultat de l'opération 2*3+5)

D ⟵ D+1; (D augmente de 1, il est **incrémenté**)

D ⟵ D-1; (D diminue de 1, il est **décrémenté**)

prix_total ⟵ nb_kg * prix_du_kg;

Si nb_kg est de type entier et prix_du_kg est de type réel alors prix_total doit être de type réel pour recevoir le résultat de l'expression

III. Concepts de base de l'algorithmique(27/34)

III.8. Les opérations d'entrées-sorties (1/3)

- Un algorithme peut avoir des interactions avec l'utilisateur
- Il peut afficher un résultat (du texte ou le contenu d'une variable) et demander à l'utilisateur de saisir une information afin de la stocker dans une variable
- En tant qu'informaticien on raisonne en se mettant **à la place de la machine, à l'aide des opérateurs de:**

III. Concepts de base de l'algorithmique(28/34)

III.8. Les opérations d'entrées-sorties (2/3)

- **De *lecture***

- L'instruction de lecture permet à l'utilisateur de saisir (au clavier) une information et de l'enregistrer en mémoire (dans la RAM)
- On utilise la commande ***Lire*** suivie entre parenthèses de la variable de type simple qui va recevoir la valeur saisie par l'utilisateur.
- **Syntaxe**

  Lire(variable1 , variable2, … , variableN)

- **Exemples**
  - **Lire**(x)
  - Cette instruction lit la valeur entrée au clavier et l'affecte à la variable x
  - **Lire**(x, y)
  - Cette instruction lit la première valeur entrée au clavier et l'affecte à x, puis lit la deuxième valeur et l'affecte à y

III. Concepts de base de l'algorithmique(29/34)

III.8. Les opérations d'entrées-sorties (3/3)

- **D'*écriture***

- L'instruction d'écriture permet d'afficher (à l'écran) une information.
- On utilise la commande ***Ecrire*** suivie entre parenthèses de la chaîne de caractères entre guillemets et/ou des variables de type simple à afficher séparées par des virgules.
- **Syntaxe**
  - **Ecrire**(expression1, expression2, ..., expressionN)
- **Exemples**
  - **Ecrire**(ch)
  - Cette instruction permet d'afficher la valeur de la variable ch à l'écran.
  - Si ch est une chaîne qui vaut "toto", cette instruction affichera toto à l'écran.
  - **Ecrire**("Bonjour!")
  - Celle-ci permet d'afficher la chaîne Bonjour! à l'écran

III. Concepts de base de l'algorithmique(30/34)

III.9. Notre premier algorithme

```
Algorithme Echange
/* Déclarations */
   variable
        x,y : entier
        tmp : entier
/* Corps du programme */
   Début
        Ecrire("Donner la valeur de l'entier x:")
        Lire(x)
        Ecrire("Donner la valeur de l'entier y:")
        Lire(y)
       Ecrire("Avant échange: x vaut", x, "et y vaut ", y)
       tmp ⟵ x
       x   ⟵ y
       y   ⟵ tmp
       Ecrire("Après échange: x vaut ", x,  "et y vaut ", y)
  Fin
```

III. Concepts de base de l'algorithmique(31/34)

III.10. Exemples d'application

1. Écrire un programme calculant la surface d'un cercle à partir de la valeur du rayon, saisie au clavier.

2. Écrire un programme qui demande à l'utilisateur de saisir le prix en francs d'une bouteille d'eau minérale de ***1,5l***. Dans un deuxième temps, le programme affiche le prix au litre, le prix d'un pack d'eau, sachant qu'il y a 6 bouteilles, dont une est offerte, ainsi que le prix d'une palette de packs d'eau, sachant qu'il y à 50 packs dans une palette dont 5 sont offerts.

III. Concepts de base de l'algorithmique(32/34)

III.11. Corrections des exemples d'application (1/3)

```
Algorithme CalculSurface
Constante
PI=3.1416
Variable
Rayon, Surface : REEL
Début
/*saisie de la valeur du rayon*/
Ecrire('Entrer la valeur du rayon')
Lire(Rayon)
/*Effectuer le calcul de la surface du cercle*/
Surface ⟵ PI*Rayon*Rayon
/*afficher le résultat*/
Ecrire('La surface est:', Surface, 'Cm2')
Fin
```

III. Concepts de base de l'algorithmique(33/34)

III.11. Corrections des exemples d'application (2/3)

Algorithme CalculPrix

Variable

PrixBouteille, PrixLitre, PrixPack : REEL

Début

/*saisir le prix d'une bouteille d'eau*/
Ecrire('Entrer le prix d'une bouteille d'eau de 1.5L')

Lire(PrixBouteille)
/*Effectuer le calcul*/
PrixLitre ⟸ (PrixBouteille*2)/3
PrixPack ⟸ PrixBouteille*5

/*afficher le résultat*/
Ecrire('Le prix au litre est de:', PrixLitre, 'FCFA')

Ecrire('Le prix d'un pack d''eau est de:', PrixPack, 'FCFA')

Ecrire('Le prix d'une palette est de:', PrixPack*45, 'FCFA')

Fin

III. Concepts de base de l'algorithmique(34/34)

III.11. Corrections des exemples d'application (3/3)

Algorithme CalculPrix

Constante

Nb_Bouteilles_Pack=6
Nb_Bouteilles_Offertes=1
Nb_Packs_Par_Palette=50
Nb_Packs_offerts=5
Nb_Bouteilles_Payantes=Nb_Bouteilles_Pack – Nb_Bouteilles_Offertes
Nb_Packs_Payants=Nb_Packs_Par_Palette – Nb_Packs_Offerts

Variable

PrixBouteille, PrixLitre, PrixPack : REEL

Début

/*saisir le prix d'une bouteille d'eau*/
Ecrire('Entrer le prix d'une bouteille d'eau de 1.5L')
Lire(PrixBouteille)

/*Effectuer le calcul*/
PrixLitre ⟵ (PrixBouteille*2)/3
PrixPack ⟵ PrixBouteille*Nb_Bouteilles_Payantes

/*afficher le résultat*/
Ecrire('Le prix au litre est de:', PrixLitre, 'FCFA')
Ecrire('Le prix d'un pack d''eau est de:', PrixPack, 'FCFA')
Ecrire('Le prix d'une palette est de:', PrixPack*Nb_Packs_Payants, 'FCFA')

Fin

FIN CHAP3