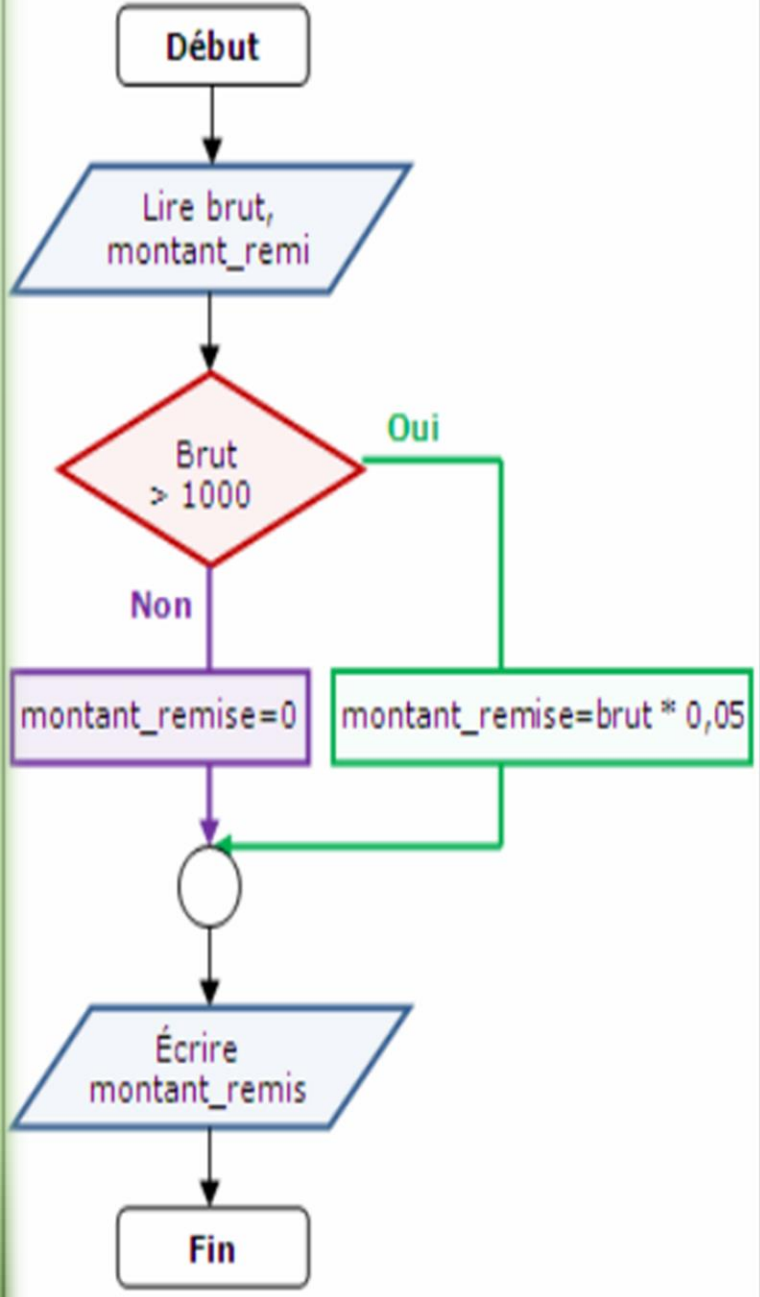
Algorithme

Organigramme	Pseudo-code
	LIRE brut, montant_remise DEBUT SI brut > 1000 ALORS montant_remise = brut * 0,05 SINON montant_remise = 0 FINSI ECRIRE montant_remise FIN

- Rappel
- Représentation d’un algorithme: Pseudo code
- Structure d’un algorithme
- Variables
- Types fondamentaux
- Constantes
- Affectation
- Instructions d’entrées/Sorties
- Expressions et opérateurs
- Exercices

Définition

Également appelé LDA (pour Langage de Description d'Algorithmes) est une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier.

(réf: wikipédia)

Structure d'un pseudo Code

- Un algorithme est composé de deux parties principales :

- La partie **déclarations** : déclarer les données qui sont utilisées par le programme.

  ↓

  Variables et constantes

- La partie **instructions ou traitement** constitue le programme principal. Les instructions sont exécutées par l'ordinateur pour transformer les données.

Structure d'un pseudo Code

Syntaxe

ALGORITHME : nom } ***Nom de l'algorithme***

CONSTANTES -- déclaration des constantes

VARIABLES **--** déclaration des variables

Partie déclaration

Instruction 1
…
Instruction n

DEBUT -- les instructions du programme

Partie traitement

Instruction 1
Instruction 2
…
Instruction n

FIN

Séquence des instructions

- L'ordre des instructions dans un algorithme est très important.
- Les opérations sont exécutées une à la suite de l'autre (De haut en bas et de gauche à droite).
- Le changement de séquence entraine le changement des résultats

Identificateurs

- Toutes entités doivent avoir un nom qui permet à l'ordinateur de les distinguer et à l'Homme de les comprendre et les distinguer.
- Ce nom est appelé **identificateur** ;
- Commence par une lettre ou un souligné (_) ;
- Est constitué par un nombre quelconque de lettres, chiffre ou soulignés;
- Doit être différents des mots clés réservés au langage de programmation;
- La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisée.
- Il faut choisir des noms significatifs aux variables, qui décrivent les données manipulées:
  ne pas utiliser: var1, var2, x, y …

Identificateurs

Exemple

VARIABLES

rayon : **Réel**
_périmètre : **Réel**

Identificateurs

- Les écritures invalides de nom de variables:

Exemple

VARIABLES

2foisLeRayon : **Réel**
Les 2 angles : **Réel**
Les.2.angles : **Réel**

Commentaires

- Les commentaires sont introduits par deux tirets pour expliquer une ligne d'instruction ou un bloc d'instructions

Exemple

```
VARIABLES    -- déclaration des variables

    rayon : Réel          -- le rayon du cercle lu au clavier
    _périmètre : Réel     -- le périmètre du cercle
```

Variables: définition

- Désigne un **emplacement mémoire** dans l'ordinateur dont le contenu peut **changer** au cours de l'exécution du programme, d'où son nom « **Variable** »

- Utilisée pour **conserver les données** qui seront manipulées par l'ordinateur.

Définition

- Caractérisée par quatre informations :

  - son **Rôle** : Cette information apparaît dans l'algorithme sous la forme d'un commentaire
  - son **nom** : permet d'identifier la variable , il est composé uniquement de lettres minuscules, majuscules, de chiffres et du caractère souligné
  - son **type** : Un type est utilisé pour caractériser l'ensemble des valeurs qu'une variable peut prendre.
  - sa **valeur** : La variable contient une information qui peut varier au cours de l'exécution d'un programme. C'est cette information que l'on appelle valeur de la variable

Comment elles fonctionnent?

- Elles sont déclarés avant leurs utilisation.
  - En déclarant une variable, on spécifie que l'on a besoin d'un espace mémoire
- Pour déclarer une variable, il faut lui donner :
  - Un nom (identificateur)
  - Un type
  - Une valeur de départ.

Types fondamentaux

- Les types de données:

Type	Domaine d'appartenance
réel	R
entier	Z
bouléen	{vrais, faux}
caractère	Type caractère(lettre minuscule, majuscule, chiffre, symbole) ▪ 'A', 'c', '3', '#'
Chaine de caractère	Phrase, suite de caractères ▪ ''Salut'', ''comment vas-tu?''

Déclaration d'une variable

Syntaxe

Variables

identificateur : type

-- Ou

liste d'identificateurs : type

Exemple

VARIABLES -- déclaration des variables

noteMath, noteFrançais, noteAnglais : **Réel** -- les notes des matières (/20)
moyenne : **Réel** -- la moyenne d'une classe (/20)
nbrEtudiant : **Entier** -- le nombre d'étudiants dans la classe
nom : **Chaîne de caractères** -- l'identifiant de la classe

Constante

- Une constante est une information manipulée par un programme et qui ne change jamais.

  par exemple : la valeur de pi , la TVA, etc.

- C'est une donnée qui n'est pas variable mais constante. Au lieu de mettre explicitement leur valeur dans le texte du programme (constantes littérales), il est préférable de leur donner un nom symbolique (et significatif). On parle alors de constantes (symboliques).

Syntaxe

Constantes

identificateur ← valeur : type

Constante

Exemple

CONSTANTES -- déclaration des constantes

PI ← 3.1415 : **Réel** -- Valeur de PI
MAJORITÉ ← 18 : **entier** -- Âge correspondant à la majorité
TVQ ← 9.975 : **Réel** -- Taux de TVQ (en %)
CAPACITÉ ← 25 : **entier** -- Nombre maximum d'étudiants dans la promotion AEC
INTITULÉ ← ''Algorithme et programmation structurée'' : Chaine de caractère
-- intitulé du cours

Constante

- Ces constantes ne peuvent pas changer au cours de l'exécution d'un programme.
- Si jamais, un changement doit être réalisé (par exemple, la valeur du TVQ soit 10%), il suffit de changer la valeur de la constante symbolique dans sa définition (et de recompiler le programme) pour que la modification soit prise en compte dans le reste de l'algorithme.

Affectation ou assignation de valeur

Définition: L'**affectation** permet de modifier la valeur associée à une variable.

Syntaxe

variable <- expression

Règle: Le type d'expression doit être **compatible** avec le type de variable.

Exemple

```
a  <--    20;
x  ←    6;
x  ←    a ;      -- signifie ``copie la valeur de a dans x'',
                 -- affecter la valeur d'une variable à une autre variable,
                    dans ce cas les types doivent être compatibles
```

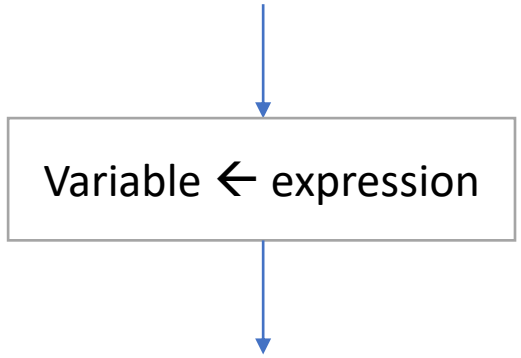
Affectation ou assignation de valeur

Évaluation : - L'**évaluation** de l'affectation se fait sur deux étapes:

1. calculer la valeur de l'expression « expression »;
2. ranger cette valeur dans la variable « variable ».

- Donc L'expression de droite est toujours évaluée avant l'affectation.

Organigramme:



Affectation ou assignation de valeur

Exercice1

Quelle est la valeur de n après l'exécution de chacune des instructions suivantes ?

1. n <- 5
2. c <- 'c'
3. Lire (n)
4. n <- 10
5. n <- n + 1

Solution

La variable n prend pour valeurs successives : 5, la valeur saisie par l'utilisateur du programme, 10 et enfin 11.
Après ces affectations les valeurs respectives de n et c sont 11 et 'c'

Affectation ou assignation de valeur

Exercice2

Donnez les valeurs des variables X, Y et Z après exécution des instructions suivantes ?

ALGORITHME: Exercice2
VARIABLES
X,Y,Z **: Entier**
DÉBUT
X ← 5
Y ← 6
Z ←0
Z ← X+Y
FIN

Que fait cet algorithme ?

Affectation ou assignation de valeur

Exercice3

Donnez les valeurs des variables X, Y et Z après exécution des instructions suivantes ?

ALGORITHME: exercice3
VARIABLES
X,Y,Z **: Entier**
DÉBUT
X ← 15
Y ← 30
X ←Y
Y ← X+2
Z ← X+Y
Z ← Y-X
FIN

Affectation ou assignation de valeur

Exercice4

Donnez les valeurs des variables X, Y après exécution des instructions suivantes ?

ALGORITHME: exercice4
VARIABLES
X,Y**: Entier**
DÉBUT
X ← 15
Y ← 30
X ←Y
Y ← X
FIN

Affectation ou assignation de valeur

Exercice

algoritme pour faire l'échange des valeurs de X et Y

ALGORITHME: exercice
VARIABLES
X,Y, Z **: Entier**

DÉBUT
X ← 15
Y ← 30
Z ←X
X ←Y
Y ← Z
FIN

Expressions

Définition: une expression est une constante, une variable ou toute combinaison d'expressions en utilisant les opérateurs arithmétiques, les opérateurs de comparaison ou les opérateurs logiques.

Exemple

```
1. 10.0          -- une constante littérale
2.  PI           -- une constante symbolique
3. rayon         -- une variable de type réel
4. 2*rayon       -- l'opérateur * appliqué sur 2 et rayon
5. 2*PI*rayon    -- une expression contenant opérateurs, constantes et variables
6. rayon >= 0    -- expression avec un opérateur de comparaison
```

Pour qu'une expression soit acceptable, il est nécessaire que les types des opérandes d'un opérateur soient compatibles.

Expressions

Règles sur la compatibilité:

- deux types égaux sont compatibles
- le type X est compatible avec le type Y si :
  - Le passage de X à Y se fait sans perte d'information
  - Autrement dit, Tout élément du type X à un correspondant dans le type Y

Exemple

Entier est **compatible** avec ***Réel***

faire l'addition d'un ***entier*** et d'un ***booléen*** n'a pas de sens.

Les **opérations logiques** ne peuvent être appliqué que sur les **expressions booléennes**.

Opérateurs

Opérateurs de comparaison

Nom	***Symbole***
Est égale à	==
Est plus petit que	<
Est plus grand que	>
Est plus petit ou égale	≤ ou <=
Est plus grand ou égale	≥ ou >=
Est diffèrent	< > ou !=

Les opérateurs arithmétiques sont:

Nom	***Symbole***
addition	+
Soustraction	-
Multiplication	*
Division entière	/
Reste de la division (juste pour les entiers)	mod
Inversion signe	-

Opérateurs

Les opérateurs logiques sont:

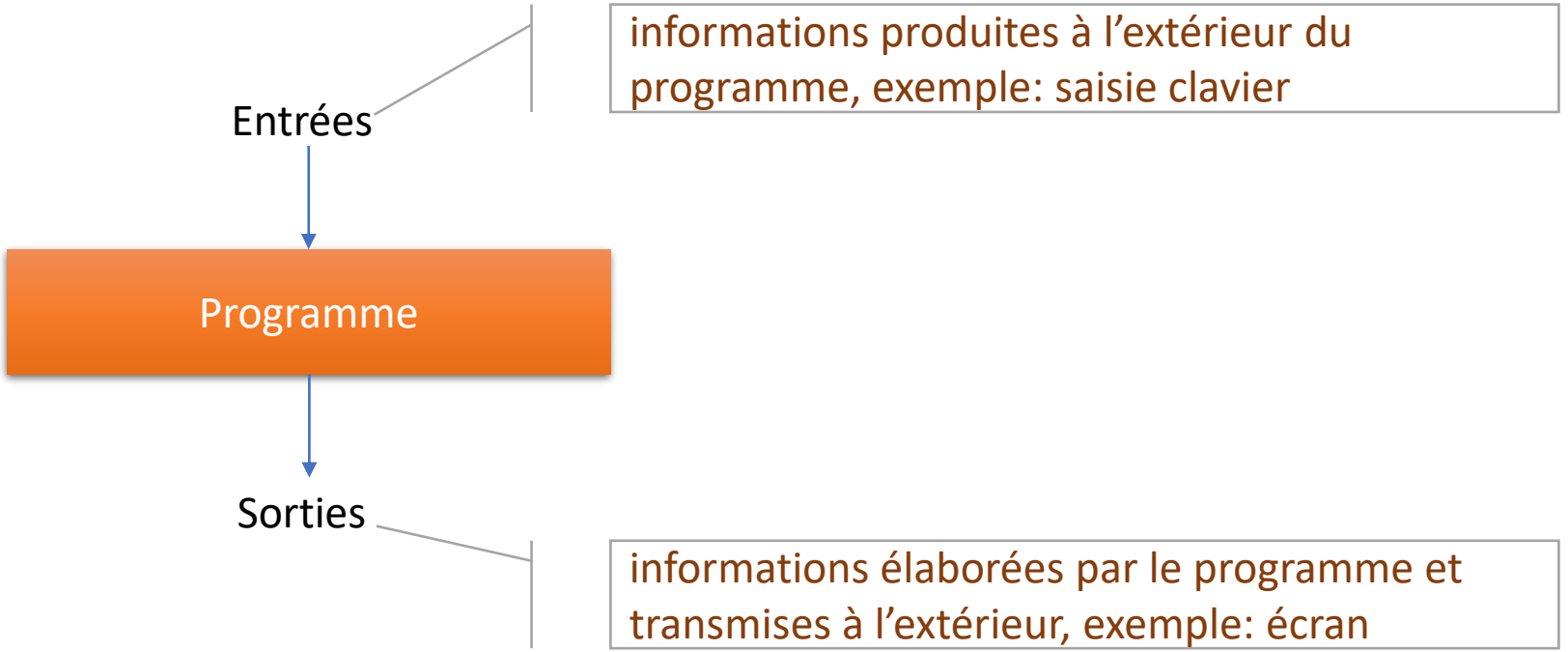
Nom	***Symbole***
conjonction	et
disjonction	ou
non	non

Priorité des opérateurs

L'ordre de priorité des opérations est le suivant :

- Les parenthèses ()
- L'exposant ^
- La multiplication ou la division * /
- Le modulo %
- Les additions et les soustractions + -

Instructions d'entrée/sorties



Opération d'entrée

Syntaxe

```
Lire(var)
```

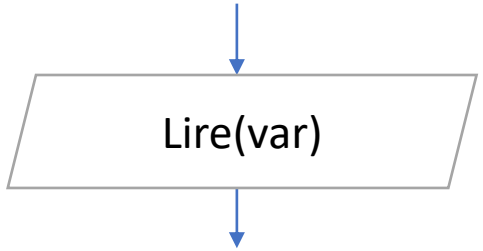
Définition: - lire sur le **périphérique d'entrée** une valeur et la ranger dans la variable ***var***
- autrement dit: affecter à la variable ***var*** la valeur lue au clavier

Règle: Le paramètre « var » est nécessairement une variable

Évaluation: sur deux étapes:

- **récupérer** l'information sur le périphérique d'entrés
- **ranger** cette information dans la variables ***var***

Organigramme:



Opération de sortie

Syntaxe

Écrire(expr)

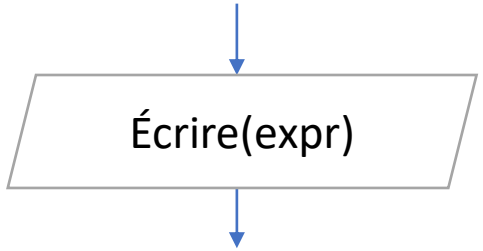
Définition: **transférer** (afficher, imprimer...) la valeur de l'expression *expr* vers le **périphérique de sortie**.

Règle: *« expr »* est une expression quelconque d'un type fondamental

Évaluation: sur deux étapes:

– **évaluer** l'expression (c'est-à-dire calculer sa valeur) ;
– **afficher** (ajouter) sur le périphérique de sortie la valeur de l'expression.

Organigramme:



Exercice5

- Écrire un algorithme qui affiche le cube d'un nombre réel saisi au clavier.

Solution : Exercice5

```
ALGORITHME : CubeDunNombre
VARIABLES     -- déclaration des variables
        nbr, resultat : Réel
DEBUT         -- les instructions du programme
        Afficher(« Entrer un nombre »)
        Lire(nbr)
        resultat ← nbr^3    -- resltat ← nbr*nbr*nbr – calcule du cube
        Afficher(« le résultat du cube : », resultat)
        Afficher(«  le résultat du carré : », nbr^2) --faire le traitement à l'affichage


FIN
```

Exercice6

Écrire un algorithme qui permet d'effectuer la saisie d'un nom, d'un prénom et affiche ensuite le nom complet.

Solution : Exercice5

```
ALGORITHME : NomPrenom
VARIABLES     -- déclaration des variables
        nom, prenom: chaine_de_caractères
DEBUT         -- les instructions du programme
        Afficherln(« Nom : »)     -- afficher et retourner à la ligne Lire(nom)
        Afficherln(« Prénom : »)  -- afficher et retourner à la ligne Lire(prenom)
        Afficher(« L'étudiant s'appelle : », nom, prenom)



FIN
```