Programme

Un programme informatique est une succession d'instructions exécutable par l'ordinateur.

Langages informatiques

a) **Un langage de programmation** est un langage informatique, permettant à un humain d'écrire un code source qui sera analysé par un ordinateur.
b) **Un langage informatique** est un **code** de communication, permettant à un être humain de dialoguer avec une machine en lui soumettant des instructions et en analysant les données matérielles fournies par le système.
c) Le langage informatique est l'intermédiaire entre le programmeur et la machine.
Exemple : un programme de résolution d'une équation du second degré
Programmation : ensemble des activités orientées vers la conception, la réalisation, le test et la maintenance de programmes.

Exemple d'un programme

PASCAL	FORTRAN
`Program somme`	`program sum`
`Var answer,x,y:real`	`implicit none`
`Begin`	`real::answer,x,y`
`Writeln('Enter two numbers:');`	`print*,'Enter two numbers'`
`Readln(x);`	`read*,x`
`Readln(y);`	`read*,y`
`answer:=x+y;`	`answer=x+y`
`Writeln('the total is:',answer);`	`print*,'the total is:',answer`
`End.`	`end program sum`

Notion d'algorithme

- Un programme informatique permet à l'ordinateur de résoudre un problème
- Avant de communiquer à l'ordinateur comment résoudre ce problème, il faut en premier lieu pouvoir le résoudre nous-même.

Un algorithme peut se comparer à une recette de cuisine

- Le résultat c'est comme le plat à cuisiner
- Les données sont l'analogue des ingrédients de la recette
- Les règles de transformations se comparent aux directives ou instructions de la recette

Algorithme informatique

Un algorithme est une suite ordonnée d'instructions qui indique la démarche à suivre pour résoudre une série de problèmes équivalents.

Exemples:

- préparer une recette de cuisine
- montrer le chemin à un touriste
- programmer un magnétoscope
- etc...

Algorithme et programme

- L'élaboration d'un algorithme précède l'étape de programmation
- Un programme est un algorithme
- Un langage de programmation est un langage compris par l'ordinateur
- L'élaboration d'un algorithme est une démarche de résolution de problème exigeante
- La rédaction d'un algorithme est un exercice de réflexion qui se fait sur papier
- L'algorithme est indépendant du langage de programmation
- Par exemple, on utilisera le même algorithme pour une implantation en Java, ou bien en C++ ou en Visual Basic
- L'algorithme est la résolution brute d'un problème informatique

Algorithmique

Algorithme = méthode de résolution
Algorithme vient du nom du célèbre mathématicien arabe **Al Khawarizmi** (Abu Ja'far Mohammed Ben Mussa Al-Khwarismi)
L'algorithmique désigne aussi la discipline qui étudie les algorithmes et leurs applications en Informatique.
L'algorithmique est la science des algorithmes.
L'algorithmique s'intéresse à l'art de construire des algorithmes ainsi qu'à caractériser leur validité, leur robustesse, leur réutilisabilité, leur complexité ou leur efficacité.

Une bonne connaissance de l'algorithmique permet d'écrire des algorithmes exacts et efficaces.

Propriétés d'un algorithme

Un algorithme doit:
- avoir un nombre fini d'étapes,
- avoir un nombre fini d'opérations par étape,
- se terminer après un nombre fini d'opérations,
- fournir un résultat.

Chaque opération doit être:
- **définie** rigoureusement et sans ambiguïté
- **effective**, c.-à-d. réalisable par une machine

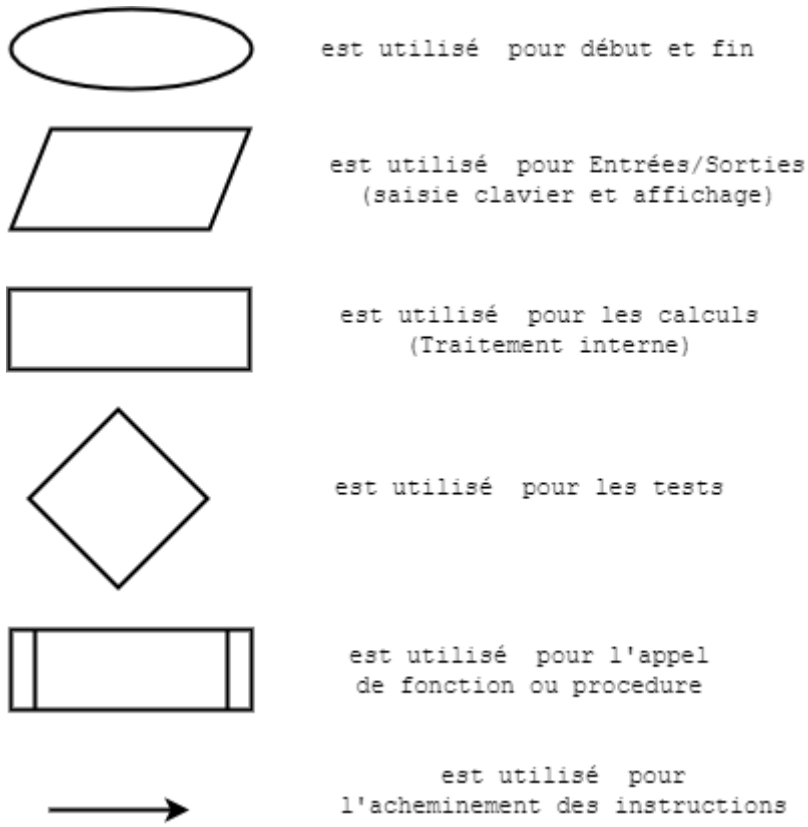
Le comportement d'un algorithme est déterministe.

Représentation d'un algorithme

Historiquement, deux façons pour représenter un algorithme:

L'Organigramme: représentation graphique avec des symboles (carrés, losanges, etc.)

- ✓ offre une vue d'ensemble de l'algorithme
- ✓ représentation quasiment abandonnée aujourd'hui



Le pseudo-code: représentation textuelle avec une série de conventions ressemblant à un langage de programmation

- ✓ plus pratique pour écrire un algorithme
- ✓ représentation largement utilisée

Structure générale d'un algorithme :

Un algorithme a la structure générale suivante :

```
ALGORITHME <identifiant_algo>

    <partie déclarations>

DEBUT

    <partie instructions>
```

FIN.

Exemple d'un algorithme

Enoncé : additionner deux nombres

Exemple :

```
Algorithme Somme ;    (1)
   Var NOMBR1, NOMBR2, SOM: réel;    (2)
Début       (3)
  Ecrire( 'donner le premier nombre :') ;    (4)
|   Lire(NOMBR1) ;        (5)
Ecrire( 'donner le deuxième nombre :') ;    (4)
|   Lire(NOMBR2) ;        (5)
|   SOM ← NOMBR1 + NOMBR2 ;     (6)
 Ecrire('la Somme =', SOM) ;     (4)

Fin.     (3)
```

Algorithmique

Notions et Instructions de base

Instructions de base

Un programme informatique est formé de quatre types d'instructions considérées comme des petites briques de base :

- ✓ l'affectation de variables
- ✓ la lecture et/ou l'écriture
- ✓ les tests
- ✓ les boucles

Notion de variable

Une variable sert à stocker la valeur d'une donnée dans un langage de programmation.

Une variable désigne un emplacement mémoire dont le contenu peut changer au cours d'un programme (d'où le nom de variable)

Chaque emplacement mémoire a un numéro qui permet d'y faire référence de façon unique : c'est ***l'adresse mémoire de cette cellule***.

Règle : La variable doit être déclarée avant d'être utilisée, ***elle doit être caractérisée par:***

- un nom (Identificateur)
- un type qui indique l'ensemble des valeurs que peut prendre la variable (entier, réel, booléen, caractère, chaîne de caractères, …)
- Une valeur

Identificateurs : règles

Le choix du nom d'une variable est soumis à quelques règles qui varient selon le langage, mais en général:

- Un nom doit commencer par une lettre alphabétique

  Exemple : E1 (1E n'est pas valide)

- doit être constitué uniquement de lettres, de chiffres et du soulignement (« _ ») (Éviter les caractères de ponctuation et les espaces)

  Exemples : SMI2008, SMI_2008

  (SMP 2008, SMP-2008, SMP;2008 : sont non valides)

- doit être différent des mots réservés du langage (par exemple en C: **int, float, double, switch, case, for, main, return**, …)

- La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisé

Conseil: pour la lisibilité du code choisir des noms significatifs qui décrivent les données manipulées

Exemples: **NoteEtudiant, Prix_TTC, Prix_HT**

Remarque: en pseudo-code algorithmique, on va respecter les règles citées, même si on est libre dans la syntaxe.

Exemple

Exemples

Identificateurs corrects:	*Identificateurs incorrects:*
`nom1`	`1nom`
`nom_2`	`nom.2`
`_nom_3`	`-nom-3`
`Nom_de_variable`	`Nom de variable`
`deuxieme_choix`	`deuxième_choix`
`mot_francais`	`mot_français`

Exercice

Quels sont les identificateurs valides ?

sucre_vanille
température
_chocolat_au_lait
convertir_en_euro
additionner_2_nombres
2par2
pet enf
pet-enf

Types des variables

Types

L'algorithmique manipule des données selon un type. Le type définit plusieurs choses :

- La nature de la donnée (par exemple, lorsqu'on manipule un type DATE, on sait que l'on a affaire à une donnée qui exprime une date, quel que soit la façon dont on la représente)
- Le format que la machine utilise pour stocker cette information. (par exemple, on peut choisir de stocker un entier sur plus ou moins d'octets).

Types de base

Un type de base "universels" est un type qui ne peut se décomposer en assemblage d'autres types. Dans la pratique, et à part certaines variations on peut dire que :

1. ***Les entiers (ENTIER)***
2. ***Les flottants (REEL)***
3. ***Les caractères (CARACTERE)***
4. ***Les booléens (BOOLEEN)***

Types entier :

Une variable est dite entière si elle prend ses valeurs dans **Z** (ensembles des nombres entiers relatifs). Ex : -5, 0 ,120 .

Déclaration : **Var** A, B, Année : **entier** ;

Types réel :

Une variable est dite réelle si elle prend ses valeurs dans **R.**

Déclaration : **Var** x, y : **réel ;**

Le type caractère :

Une variable est dite caractère si sa valeur est un caractère quelconque. Un caractère peut appartenir au domaine des chiffres de ‘0’ à ‘9’, des lettres de ‘A’ à ‘Z’ (majuscules ou minuscules) et des caractères spéciaux (‘+’ ‘-‘ ’, ‘ ’;’ ‘.’ ‘(‘ ‘{‘ ‘[‘ ‘%’...).

Déclaration : **Var** Nom, prénom : **caractère** ;

Le type logique (booléen) :

Une variable de type logique a une valeur logique (ou booléenne) qui prend l'une des deux valeurs '***vrai***' ou '***faux***'. Elle intervient dans l'évaluation d'une condition.

Déclaration : **Var** Max : **booléen** ;

Exemple

définition	*nom de variable*	*type*	*exemple de valeur*
le solde initial d'un compte bancaire	**SOLDE**	**réel**	895.25
le nombre de personnes d'une enquête	**NBPERS**	**entier**	1022
une phrase à analyser	**PHRASE**	**chaîne de caractère**	C'est une phrase !
une lettre à rechercher	**LETTR**	**caractère**	e
l'état d'une comparaison (A=B)	**COMPAR**	**booléen**	vrai

Déclaration des variables

Rappel: toute variable utilisée dans un programme doit avoir fait l'objet d'une déclaration préalable.
Cette déclaration est effectuée en utilisant le mot réservé variable ou bien var, elle est comme suit :

```
variable
<nom-de-variable1> :<type1>
<nom-de-variable>:<type2>
```

Exemple: **Variables i,j,k : entier**

```
x, y : réel
OK: booléen
Ch1, ch2 : chaîne de caractères
```

Exemple en Fortran

```
INTEGER:: Zip,Total,counter
REAL:: Average,X,Difference
LOGICAL:: condition,OK
```

Remarques

- Pour le type numérique, on va se limiter aux entiers et réels sans considérer les sous types
- Pour chaque type de variables, il existe un ensemble d'opérations correspondant.
- Une variable est l'association d'un nom avec un type, permettant de mémoriser une valeur de ce type.

Constante

Une constante est une variable dont la valeur ne change pas au cours de l'exécution du programme, elle peut être un nombre, un caractère, ou une chaine de caractères.

En pseudo-code :

```
Constante
<nom-de-constante1>=<valeur1>
<nom-de-constante2>=<valeur2>
```

(Par convention, les noms de constantes sont en majuscules)

Exemple : pour calculer la surface des cercles, la valeur de pi est une constante mais le rayon est une variable.

Constante PI=3.14 : réel, MAXI=32 : entier

Une constante doit toujours recevoir une valeur dès sa déclaration.

Exemple en Fortran

```
INTEGER, PARAMETER :: MAXIMUM = 10
REAL, PARAMETER :: PI = 3.1415926, E = 2.17828
LOGICAL, PARAMETER :: TRUE = .true., FALSE = .false.
```

Affectation (الإسناد)

L'affectation consiste à attribuer une valeur à une variable (c'est-à-dire remplir ou modifier le contenu d'une zone mémoire)

En pseudo-code, l'affectation est notée par le signe ←

Var← e : attribue la valeur de e à la variable Var

- e peut être une valeur, une autre variable ou une expression
- Var et e doivent être de même type ou de types compatibles
- l'affectation ne modifie que ce qui est à gauche de la flèche

Exemples :

```
i ←1
j ←i
k ←i+j
x ←10.3
OK ←FAUX
ch1 ←"SMI"
ch2 ←ch1
x ←4
x ←j
```

(avec i, j, k : entier; x :réel; ok :booléen; ch1,ch2 :chaine de caractères)

Exemple 1

```
Variable A en Numérique
Début
A ← 12
Fin
```

Exemple 2

```
Variable A en Numérique
Début
A ← 34
Fin
```

Exemples **non valides: i ←10.3 OK ←"SMI" j ←x**

Les langages de programmation C, C++, Java, utilisent le signe égal = pour l'affectation ←.

Remarques:

- ☞ Lors d'une affectation, l'expression de droite est évaluée et la valeur trouvée est affectée à la variable de gauche. Ainsi, A←B est différente de B←A
- ☞ l'affectation est différente d'une équation mathématique
- ☞ Les opérations x ← x+1 et x ← x-1 ont un sens en programmation et se nomment respectivement incrémentation et décrémentation.
- ☞ A+1 ← 3 n'est pas possible en langages de programmation et n'est pas équivalente à A ← 2

☞ Certains langages donnent des valeurs par défaut aux variables déclarées. Pour éviter tout problème il est préférable **d'initialiser les variables** déclarées.

Affectation exercices

Donnez les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C: Entier
Début
A ← 7
B ← 17
A ← B
B ← A+5
C ← A + B
C ← B – A
Fin
```

Corrigé

Après	La valeur des variables est :		
A ← 7	A = 7	B = ?	C = ?
B ← 17	A =7	B = 17	C = ?
A ← B	A =17	B = 17	C = ?
B ← A+5	A =17	B =22	C = ?
C ← A+B	A = 17	B = 22	C = 39
C ← B-A	**A = 17**	**B = 22**	**C = 5**

Les actions d'Entrée/Sortie

Imaginons que nous ayons fait un programme pour calculer le carré d'un nombre, mettons 12. Si on a fait au plus simple, on a écrit un truc du genre :

```
Variable A en Numérique

Début
A ← 12^2

Fin
```

D'une part, ce programme nous donne le carré de 12. Mais si l'on veut le carré d'un autre nombre que 12, il faut réécrire le programme.

C'est pourquoi, heureusement, il existe des d'instructions pour permettre à la machine de dialoguer avec l'utilisateur.

Dans un sens, ces instructions permettent à l'utilisateur de rentrer des valeurs au clavier pour qu'elles soient utilisées par le programme. Cette opération est **la lecture**.

Dans l'autre sens, d'autres instructions permettent au programme de communiquer des valeurs à l'utilisateur en les affichant à l'écran. Cette opération est **l'écriture**.

Les instructions de lecture et d'écriture

La lecture:

C'est l'action par laquelle nous pouvons introduire des données en utilisant le clavier. Elle est notée par :

```
Lire (x)
```

Lire (Ident1,Ident2,…,identN) **;**

Ce qui signifie mettre dans la zone mémoire x la donnée tapée sur le clavier.

L'écriture:

C'est l'action par laquelle nous pouvons communiquer un résultat ou un message à l'utilisateur par l'intermédiaire de l'écran. Elle est notée par:

Écrire (E)

Ecrire (Expression1, Expression2, Expression3…, ExpressionN) ;

Ce qui signifie évaluer l'expression E et afficher le résultat.

Exemple

Ecrire ("Entrez votre nom : ") ;

Lire (Nom) ;

Ecrire ("Entrez votre mot de passe : ") ;

Lire (Mot) ;

Ecrire ("Bonjour ", Nom) ;

Exercices

1.

```
Quel résultat produit le programme suivant ?
Variables val, double numériques
Début
Val ← 231
Double ← Val * 2
Ecrire Val
Ecrire Double
Fin

Solution
```

On verra apparaître à l'écran 231, puis 462 (qui vaut 231 * 2)

2.

Ecrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

```
Solution
Variables nb, carr en Entier
Début
Ecrire "Entrez un nombre :"
Lire nb
carr ← nb * nb
Ecrire "Son carré est : ", carr
Fin
```

En fait, on pourrait tout aussi bien économiser la variable carr en remplaçant les deux avant-dernières lignes par :

```
Ecrire "Son carré est : ", nb*nb
```

C'est une question de style ; dans un cas, on privilégie la lisibilité de l'algorithme, dans l'autre, on privilégie l'économie d'une variable.

Travaux Dirigés

Exercice 1

Quelles seront les valeurs des variables après exécution des instructions suivantes ?

```
1-Algorithme exemple1
Variables A, B : Entier
Début
A ← 1
B ← A + 3
A ← 3
```

```
Fin.
2-Algorithme exemple2
Variables A, B : Entier
Début
A ← 5
B ← A + 4
A ← A + 1
B ← A – 4
Fin.
3-Algorithme exemple3
Variables A, B en Entier
Début
A ← 5
B ← 2
A ← B
B ← A
Fin.
4-Algorithme exemple4
Variables A, B, C : Entier
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B – A
Fin.

5-Algorithme exemple5
Variables A, B, C: Entier
Début
A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C
Fin.

6-Algorithme exemple6
Variable A,B : entier
Début
A ← 5
B ← 2
A ← B
B ← A
 Fin

7-Algorithme exemple7
Variable A,B,C : entier
Début
A ← 5
B ← 7
B ← A
A ← B
Fin
```

Exercice 2

Quel résultat produit le programme suivant ?

```
Variable val, double : réel

Début
Val ← 231
Double ← Val * 2
Ecrire(val)
Ecrire (Double)
Fin
```

Exercice 3

Ecrire un programme qui demande un nombre à l'utilisateur, puis calcule et affiche le carré de ce nombre.

Exercice 4

Ecrire un algorithme qui lit 3 nombres réels et calcule leur somme.

Exercice 5

Ecrire un algorithme qui permet d'échanger 2 nombres entiers.

Exercice 6

On dispose de trois variables A, B et C. Écrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

Exercice 7

Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui fournit le prix total TTC correspondant.

Exercice 8

Ecrire un algorithme qui convertit en Octets, Kilo octets, Mega octets et Giga octets un nombre donné en bits.