Algorithmique

Cours réalisé par Dr. Mohamed RAMDANI

Établissement: Université Mohamed Khider Biskra

Faculté: Sciences Exactes et Sciences de la Nature et de la Vie

Département: Informatique

Mail: mohamed.ramdani@univ-biskra.dz

Avril 2022

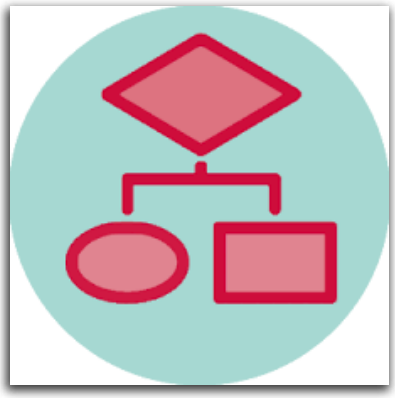


Table des matières

Chapitre 02: Algorithme Simple Séquentiel

1. Les notions de langage et langage algorithmique

Un langage *Définition*

c'est la capacité d'exprimer une pensée et de communiquer au moyen d'un système de signes (par voix, gestes ou graphiques)

- Les signes ont une sémantique
- Le langage a souvent un **lexique** et une **syntaxe**

Un langage algorithmique *Définition*

c'est la capacité d'exprimer et de décrire une solution d'un problème au moyen d'un ensemble de règles (par des mots

réservés, des mots ordinaires, des caractères) en suivant une organisation d'expression. Un langage algorithmique permet d'écrire un Algorithme

2. Les parties d'un algorithme

Un algorithme commence par un entête qui début par le mot réservé: **Algorithme**

En suite, on mentionne le nom de l'algorithme qui est donné par le concepteur (celui qui est en train de concevoir la solution

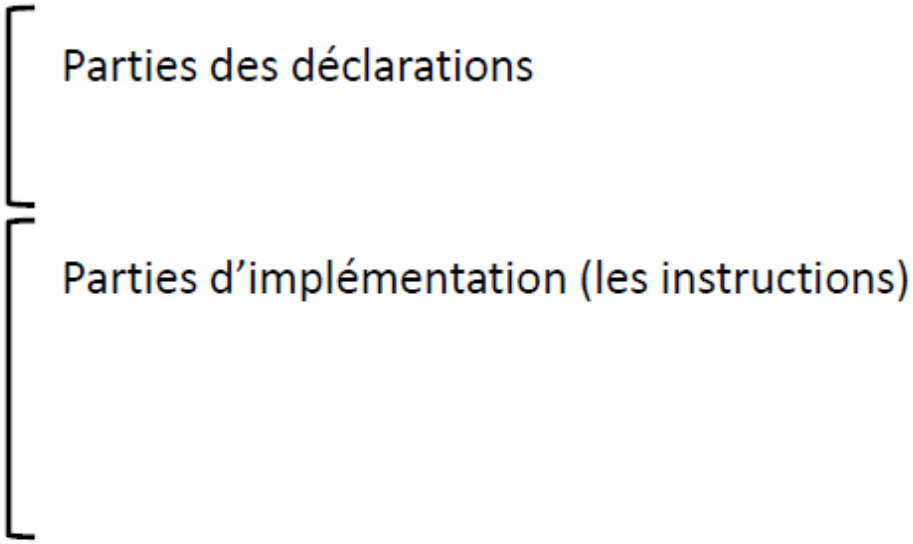
algorithmique). Cet entête peut être terminé par un point virgule

Exemple

```
Algorithme name_algo ;
```

Fondamental

Par la suite, on conçoit deux parties (compartiments) :



Les parties d'un Algorithme

Méthode

La partie des déclarations: dans cette partie:

1. On déclare toutes les données utilisées dans l'algorithme (les entrées, les sorties et les données intermédiaires); on distingue deux types de données: les variables et les constantes.
2. On définit les nouveaux types de données (comme données non ordinaires et hétérogènes: ex. les enregistrements.

La partie d'Implémentation: dans cette partie:

On détermine toutes les instructions à exécuter et à suivre pour arriver à la solution du problème visé

3. Les données: les variables et les constantes

Définition

une donnée est la représentation d'un objet dans un algorithme ou un programme (un code source) ou en mémoire, Lorsque la donnée aura un sens, elle est appelée une

information. Si **l'information** permet d'aboutir à une action, elle est appelée une connaissance, la connaissance est acquise par l'expérience.

Une variable *Définition*

est la représentation abstraire d'un objet comme un chiffre ou une lettre ou suite de lettre ou

autres.

- Le contenu (ou la valeur) de cet objet peut être modifié par une instruction (durant l'exécution du programme).
- Pour utiliser une variable, on doit la déclarer (dans la partie déclaration) par:
    1. Donner un **nom** (à cette variable)
    2. Préciser son **type**

déclaration d'une variable dans un programme signifie la réservation d'un espace mémoire pour cette variable

- L'adresse de cet espace est le nom de la variable
- La taille de cet espace dépendra du type de la variable (des exemples de types de données vont être expliqués prochainement)
- La valeur de la variable est introduite (donnée) durant l'exécution :
    1. soit par l'utilisateur (en tapant sur le clavier)
    2. ou bien par une instruction

Exemple

Algorithme Name_Algo;

Var

name_variable1: **type_variable1**;

name_variable2: **type_variable2**;

...

name_variableN: **type_variableN**;

Remarque

En Algorithmique ou en C, il possible de déclarer plusieurs

variables du même type dans la même ligne:

Ex. (en algorithmique):

Var

name_variable1, name_variable2 : type_variable;

Ex. (en C):

int i, j;

- En C, il possible d'initialiser la variable durant la déclaration (lui donner une valeur initiale) :

Ex.

int i = 9;

Une constante ***Définition***

est la représentation abstraire d'un objet dont le contenu (ou la valeur) de cet objet ne peut pas être modifié par une instruction (durant l'exécution du programme).

Pour utiliser une constante, on doit la déclarer (dans la partie déclaration) par:

1. Donner un **nom** (à cette constante)
2. Préciser sa **valeur** (on appelle cette précision: **Initialisation**)

Remarque

1. En algorithmique, on ne précise pas le type de la constante.
2. En langage C, on peut préciser le type de la constante.

Exemple

Algorithme Name_Algo;
Const
name_constante1 = **valeur_constante1**;
name_constante2 = **valeur_constante2**;
...
name_constanteN = **valeur_constanteN**;
Var
name_variable1: **type_variable1**;

En langage C

Exemple

```
#include <stdio.h>
void main()
{
const float Pi = 3.14;
const int Max = 50;
const Min = -50;
...
int x;
...}
```

Auto-Formation

vidéo youtube

[cf. youtube]

4. Les types de données

Avant d'utiliser une variable, on doit indiquer son type (sa nature), ce qui précise:

1. toutes les valeurs que peut prendre la variable ;
2. La taille de l'espace de stockage réservé à cette variable (en octet. 1 octet = 8 bits)
3. Les opérations qu'il est possible d'effectuer avec la variable (ex. pour un caractère, il n'est pas possible de réaliser une division).

Pour préciser le type d'une variable, il est nécessaire d'utiliser un mot-clé spécifique

Rappel

- Indication de type de variable en Algorithmique:

Algorithme Name_Algo;

Var

name_variable1: type_variable1;

...

- • Indication de type de variable en C :

#include <stdio.h>

void main()

{

type_variable1 name_variable1;

...}

Fondamental

Les différents types simple en Algorithmique:

- **Entier : c'est nombre entier**
- **Réel : c'est nombre réel**
- **Caractère : c'est nombre caractère (A, a, b, @, ?, ...)**
- **Booléen : c'est nombre logique (0 ou 1)**

Les différents types simple en C

Définition	Description	Valeur Min	Valeur Max	Nombre d'octets
int	Entier standard	-32768	32767	2
long	Entier Long	-2147483648	2147483648	4
unsigned int	Entier sans signé	0	65535	2
float	Réel	3.4 * 10-38	3.4 * 10+38	4
double	Réel long	1.7 * 10-308	1.7 * 10+308	8
char	Caractère	-127	127	1
bool	Booléen	0	1	1

5. Les opérations de base

En algorithmique et en C, on définit un ensemble d'opérations de base qui

sont des opérations arithmétiques et logiques et de comparaison :

- **Les opérations arithmétique**s : c'est des instructions qui permettent d'effectuer un calcul mathématique (formulées comme suit):

+ (addition):

En Algorithmique: Ex. A + B

En C: Ex. A + B

- (soustraction):

En Algorithmique: Ex. A - B

En C: Ex. A * B

*** (produit):**

En Algorithmique: Ex. A * B

En C: Ex. A * B

/ (division):

En Algorithmique, on distingue entre la division entière

(symbolisée: div) Ex. A div B

et la division (symbolisée: /).

Ex. A / B

En C, le utilise seulement le symbole (/) pour les deux types de

division (le compilateur fait la différence selon le type des opérandes –

les chiffres impliqués dans la division-) Ex. A / B

mod (modulo): (le reste de la division entière) c'est une opération qui

permet de donner le reste d'une division entière

En Algorithmique: Ex.

- **Les opérations logiques** : c'est des instructions qui permettent de vérifier si

une ou plusieurs conditions sont vraies (formulées comme suit):

L'opération (et):

En Algorithmique: Ex. A et B

En C: Ex. A && B

L'opération (ou):

En Algorithmique: Ex. A ou B

En C: Ex. A || B

L'opération (non logique):

En Algorithmique: Ex. NON (A)

En C: Ex. ! (A)

- **Les opérations de comparaison:** c'est des instructions qui permettent de vérifier si

une ou plusieurs comparaisons sont vraies (formulées comme suit):

Opérateur d'égalité:

En Algorithmique: Ex. A = B

En C: Ex. A == B (ne pas confondre avec le symbole = l'affectation qu'on va

voir dans la sous section 2.6)

Opération d'infériorité:

En Algorithmique: (infériorité stricte Ex. A < B) (infériorité Ex. A <= B)

En C: (infériorité stricte Ex. A < B) (infériorité Ex. A <= B)

Opération de supériorité:

En Algorithmique: (supériorité stricte Ex. A > B) (supériorité Ex. A >= B)

En C: (supériorité stricte Ex. A > B) (supériorité Ex. A >= B)

L'opération de différence:

En Algorithmique:

Ex. A <> B

En C: Ex. A != B

Auto-Formation

vidéo youtube

[cf. youtube1]

6. Les instruction de base

L'affectation ***Définition***

C'est une instruction qui permet d'insérer une valeur à une variable au cours de l'exécution du programme. Cette valeur doit avoir le même type que la variable

- En algorithmique, une affectation est écrite comme suit:

Variable <-- Valeur ;

Ex.

x<-- 12 ;

On dit que la variable x reçoit la valeur 12

- En C, une affectation est programmée comme suit:

Variable = Valeur ;

Ex.

x = 12 ;

Les opérations d'entrées /sorties ***Définition***

Opération d'entrée: une opération qui permet de lire des données tapées au clavier

Pour lire la valeur de la variable A à partir du clavier, On écrit:

En algorithmique:

Lire (A);

Ici, la valeur tapée au clavier par l'utilisateur est transférée à l'espace

réservé pour la variable A

Complément

En C:

Pour les entrées, on utilise la fonction scanf qui permet de lire les informations tapées au clavier par l'utilisateur selon un certain format.

- On écrit: scanf("format", &varia) ;

1. format: représente une suite de codes pour exprimer le format/type de la variable à lire
2. &varia: est l'adresse de l'espace mémoire réservée à la variable varia pour qu'elle soit utilisée (l'adresse) pour ranger la valeur lue au clavier.

Exemple :

scanf("%d", &i); ici, i est une variable de type entier (int)

Les opérations d'entrées /sorties

Définition

Opération de Sortie: une opération qui permet d'afficher sur l'écran :

- Soit un commentaire
- Ou bien la valeur d'une donnée (qui est déjà utilisée dans l'algorithme/programme).

• On écrit:

• En algorithmique:

- Pour un commentaire:

Écrire ("commentaire... ");

- Pour la valeur de donnée A:

Écrire (A);

- On peut utiliser une seule commande Écrire pour afficher des commentaires et des valeurs de données (à la fois):

Écrire ("commentaire1", A, "commentaire1", B);

Complément

En C:

On utilise la fonction printf qui permet d'afficher sur écran un commentaire et ou des valeurs de données enregistrées dans la

mémoire centrale, et ceci selon le format adéquat de la donnée.

On écrit:

1. Pour un commentaire: printf("commentaire") ;
2. Pour une valeur de donnée: printf("format", varia) ;

Exemples :

printf("Bonjour");

printf("%d", i);

Auto-Formation

vidéo youtube

[cf. youtube2]

7. Construction d'un algorithme simple

On écrit un algorithme qui permet l'addition de deux entiers:

Algorithme Addition_2Entiers;

Var

A, B, C: **entier**;

Début

Écrire("Entrer deux entiers :");

Lire(A, B);

C <--A + B;

Écrire("La somme de ", A, " et ", B, " est: ", C);

Fin.

8. Exercice : Quiz List: Notion d'un Algorithme

Exercice

Qu'est-ce qu'un algorithme ?

- Un problème de décision
- Une méthode ou un procédé décrit pas à pas
- Un langage de programmation
- Un code numérique

Exercice

Qu'y a-t-il de commun entre une recette de cuisine et un algorithme ?

- On peut les appliquer comme des théorèmes
- Aucun point commun, une recette n'a rien de mathématique
- Il s'agit d'une suite d'instructions
- Appliquez-les et dans tous les cas, vous obtiendrez un bon résultat