Université : A.Mira – Bejaia
Faculté : Sciences Exactes
Département : Informatique

Année Universitaire : 2022-2023

Niveau : Première Année Ingénieur

ALSD-1 - Série de TD N°01

Sommaire

Adapté par: Redouane OUZEGGANE
rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz

TD ALSD-1

SÉRIE DE TD N°01 (LIRE, ÉCRIRE & AFFECTATION)

Exercice 01

1) C'est quoi un algorithme ?
2) C'est quoi le rôle des variables et des instructions dans un algorithme ?
3) Quelle sont les étapes d'une démarche de résolution algorithmique d'un problème ?
4) Indiquer le schéma générale d'un algorithme ?
5) C'est quoi un identificateur ?

Exercice 02

1- Écrire un algorithme qui permet de lire deux valeurs entières a et b, de calculer et afficher la somme de ces deux valeurs ? Il faut suivre le étapes suivantes :

- Indiquer les variables de l'algorithme : entrées, sorties et intermédiaire
- Schématiser l'algorithme
- Spécifier les instructions de la partie traitement
- Écrire l'algorithme

2- Dérouler l'algorithme de la question 1 ?

3- Traduire l'algorithme en programme C ?

Exercice 03 (Supplémentaire)

Écrire un algorithme, puis traduit le en programme C (éventuellement PASCAL), pour chacun des problèmes suivants :

1) permuter entre les deux variables X et Y ?
2) permuter entre les trois variables X, Y et Z de telle sorte que la valeur de X soit dans Y, celle de Y dans Z et la valeur de Z dans X ?
3) calculer la division entre deux nombres réels a et b ?
4) calculer la division euclidienne de deux nombres entiers a et b ?
5) calculer la somme de a et b et le produit de b et c ?
6) calculer la valeur absolue, le carré et la racine carrée d'un nombre ?

Remarque : Réaliser le déroulement de chaque algorithme avec des valeurs arbitraire.

Solution

EXERCICE N°01 : QUESTIONS DIVERSES

1) C'est quoi un algorithme ?

Un algorithme est une suite d'instructions, logiquement ordonnées, qui permettent de transformer des données en entrée à des données de sorties. Les données de sortie représente une solution à un problème donnée.

2) C'est quoi le rôle des variables et des instructions dans un algorithme ?

- Les variable permettent de représenter des objets (concrets ou abstrait) du monde réel. Les variables (espaces mémoires), sont utilisées pour introduire des valeurs (par l'utilisateur), calculer de nouvelles valeurs à travers des formules, et contenir le résultat de la solution.
- Les instructions permettent de manipuler les variables (données) dans un algorithme :
  - Introduire des valeurs
  - Calculer des nouvelles valeurs (à travers des formules mathématiques)
  - Affichage des résultat
  - ...

3) Quelle sont les étapes d'une démarche de résolution algorithmique d'un problème ?

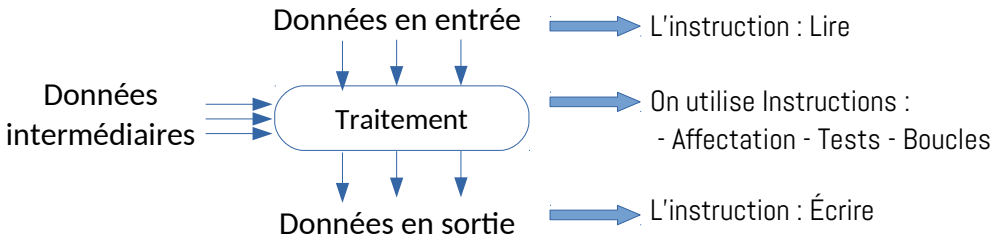
Pour écrire un algorithme, nous suivons les étapes suivantes :

1- Analyser le problème : Données, données en entrées, en sorties, les formules, ...
2- Regrouper les éléments de l'analyse pour écrire l'algorithme
3- Valider l'algorithme par déroulement (si c'est possible)
4- Traduire l'algorithme à un programme
5- Exécuter le programme pour obtenir des résultats
6- Vérifier si les résultats sont valides. Si les résultats sont invalides, nous vérifions les étapes précédentes.

4) Indiquer le schéma générale d'un algorithme

Un algorithme est constitué des données et des instructions. Pour les données nous avons des données d'entrée, des données de sortie et, éventuellement, des données intermédiaire. Les données en entrée sont saisies par l'instruction de lecture (Lire), les données de sortie sont affichées par l'instruction d'écriture (Écrire), et les données intermédiaires sont manipulées par les autres instructions (affectation, tests, boucles).

Un algorithme peut être schématisé comme suit :



Remarque :

Dans quelques cas, on trouve une instruction d'entrée qui fait partie de traitement, et aussi des instructions de sortie qui font partie de la partie traitement.

5) C'est quoi un identificateur

Un identificateur est un nom unique qu'on utilise pour désigner un objet (variable, constantes, procédure, fonction, ...) dans un algorithme. Chaque données, soit constante ou variable, doit avoir un identificateur (pour la déclarer et l'utiliser par des instruction).

Syntaxiquement parlant, un identificateur est une chaîne de caractères qui vérifier les trois conditions suivantes :

1- Contient uniquement des caractères alphabétiques, numériques et trait souligné (_ : tiré 8) : Donc, nous avons 38 caractères possibles à utiliser. On dit : un identificateur est une chaîne de caractères alpha-numérique.

2- Ne doit pas commencer par un caractère numérique.

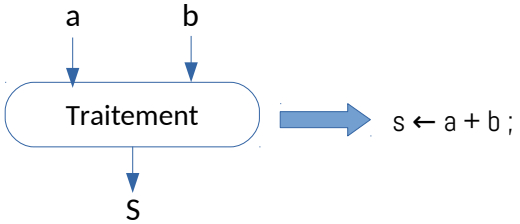
3- Ne doit pas appartenir aux mots réservé du langage de programmation utilisé, aussi, éviter les mots clés de l'algorithmique, les noms des fonctions et procédures standard. Par exemple, les mots suivants ne doivent pas être utilisés comme identificateur : *main*, *scanf*, *printf*, *algorithme*, *début*, *fin*, *si*, *sinon*, *pour*, *fin-pour*, ...

EXERCICE N°02 : ALGORITHME DE SOMME

1) Écrire un algorithme de somme

Nous voulons écrire l'algorithme qui permet de calculer la somme de deux valeurs entières *a* et *b*. Avant d'écrire l'algorithme, nous suivons des étapes pour analyser la question (le problème : très simple question) :

- Les variable du problème sont : a, b, s. Tel-que s = a+b ;
- Les variables d'entrée : *a* et *b*
- Les variables de sortie : *s*.
- La formule *s = a+b* devient en algorithmique : *s ← a+b;*
- Le schéma d'E/S (Entrée/Sortie) de l'algorithme sera :



En regroupant les éléments d'analyse ci-dessous, nous aurons l'algorithme suivant :

Algorithme

Algorithme Exemple;
Variables
a, b, s : entier;
Debut
{Entrées}
Lire(a, b) ;

{Traitement}
S ← a + b;

{Sorties}
Écrire(S) ;
Fin.

Avec les messages

Algorithme

Algorithme Exemple;
Variables
a, b, s : entier;
Debut
{Entrées}
Écrire('Donner deux valeurs a et b :') ;
Lire(a, b) ;

{Traitement}
S ← a + b;

{Sorties}
Écrire('La Somme S=a+b est :', S) ;
Fin.

2- Dérouler l'algorithme

Pour dérouler un algorithme, nous utilisons un tableau à trois colonnes :

- 1ère colonne : Instructions
- 2ème colonne : Variable : cette colonnes est divisée en sous-colonnes, chacune d'elle est destinée à une variable.
- 3ème colonne : Affichage (On peut ne pas utiliser cette colonne, voir ci-dessous) :

Instructions	Variables			Affichage
	a	b	S	
Écrire('Donner deux valeurs a et b :') ;	/	/	/	**Donner deux valeurs a et b : _**
Lire(a, b) ;	13	25	/	
S ← a+b	″	″	38	
Écrire('La Somme S=a+b est :', S) ;	″	″	″	**La somme S=a+b est : 38**

Voici un autre tableau de déroulement :

Instructions	Variables		
	a	b	S
Écrire('Donner deux valeurs a et b :') ;	**Donner deux valeurs a et b : _**		
Lire(a, b) ;	13	25	/
S ← a+b	″	″	38
Écrire('La Somme S=a+b est :', S) ;	**La somme S = a+b est : 38**		

Un autre déroulement sans messages :

Instructions	Variables		
	a	b	S
Lire(a, b) ;	13	25	/
S ← a+b	″	″	38
Écrire(S) ;			38

3- Traduire l'algorithme en Programme C

Programme C

```c
#include <stdio.h>

int main()
{
    int a, b, S;
    /* Entrées */
    printf("Donner deux valeurs entières a et b : ");
    scanf("%d %d", &a, &b);

    /* Traitement */
   S = a+b;

    /* Sortie */
   printf("La somme S=a+b est : %d", S);

   return 0;
}
```

Vous pouvez exécuter ce programme, en ligne, via le lien suivant : https://onlinegdb.com/lzf3tKH3I

Un exemple d'exécution du programme ci-dessus :

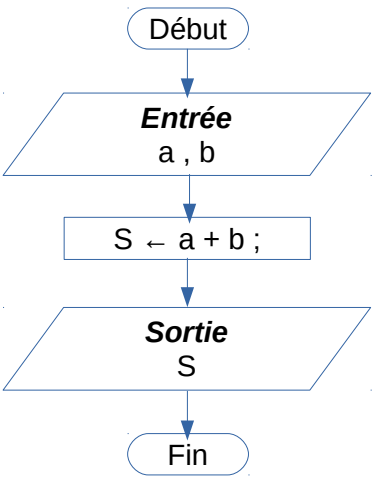
```
Donner deux valeurs entières a et b : 13 25
La somme S=a+b est : 38

...Program finished with exit code 0
Press ENTER to exit console.
```

4- L'organigramme (Représentation graphique)

Voici l'organigramme (logigramme) de l'algorithme :

Un organigramme est une représentation graphique de l'algorithme.



1- Permuter entre X et Y

Analyse du problème :

On prends un exemple, pour illustrer (et clarifier) le problème :

soit x = 65 et y = 17, à la fin de l'algorithme, nous devons avoir x = 17 et y=65

Variables d'entrée : x et y (la première instruction : lire(x,y);)
Variables de sortie : x et y (la dernière instruction : écrire(x,y);)
Traitement : échanger les valeurs de x et y

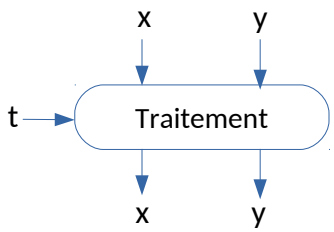
La solution, triviale, qui vient à l'esprit, est :

x ← y ;
y ←x ;

Le problème ici, est que la valeur initiale de x sera perdue, comme suit :

Instructions	***Variables***	
	x	**y**
Lire (x, y)	65	17
x ←y	17	17
y ←t	17	17
Écrire (x,y)	**17**	**17**

Le problème, avec cette solution, et que la valeur initiale de x est perdue (écrasée), comme montré dans le déroulement précédent.



L'idée est d'utiliser une troisième variable pour sauvegarder la valeur de x avant qu'elle soit écrasée. Pour échanger (permuter) les deux variable de x et y, on utilise une troisième variable t qui permet de conserver, par exemple la valeur de x, comme suit :

t ← x ; x ← y; y ← t ; *{t contient l'ancienne valeur de x}*

Le lien du code : https://onlinegdb.com/NUZMEOOrT

Algorithme

```
Algorithme Exo2_1;
  Variables
      x, y, t : entier;
Début
    {Entrées}
    Lire(x, y) ;
    {Traitement}
    t ← x;
    x ← y;
    y ← t ;
    {Sorties}
    Écrire(x, y) ;
Fin.
```

Programme C

```c
#include <stdio.h>

int main()
{
    int x, y, t;
    /* Entrées */
    printf("Donner deux valeurs entières X et Y : ");
    scanf("%d %d", &x, &y);

    /* Traitement */
   t = x;
   x = y;
   y = t;

    /* Sortie */
   printf("X = %d   et   Y = %d", x, y);

   return 0;
}
```

Exemple de déroulement pour x = -20 et y=108 :

Instructions	***Variables***		
	x	**y**	**t**
Lire (x, y)	-20	108	/
t ← x	"	"	-20
x ←y	108	"	"
y ←t	108	-20	"
Écrire (x,y)	108	-20	"

2 – Permuter entre les trois variables X , Y et Z ?

Analyse du problème :

Dans la permutation entre trois variables, X, Y et Z, nous aurons plusieurs cas, possibles. Pour cela, dans l'énoncé de la question N° 02, il y a une précision dans le sens de permutation :

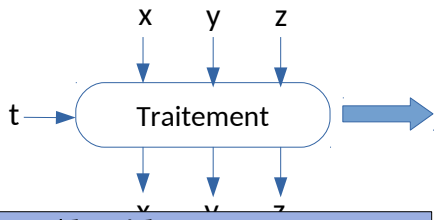
la valeur de X soit dans Y, celle de Y dans Z et la valeur de Z dans X

C'est-à-dire :

Y ← X ; X ← Z ; Z←Y ;

De la même façon que le problème précédent, nous devons utiliser une 4ème variable T pour sauvegarder la valeur de Y (puisque Y est le premier à modifier dans la première affectation). Ainsi la solution devient :

T ← Y ; Y←X ; X←Z ; Z←T; *{T contient l'ancienne valeur de Y avant qu'elle soit modifiée}*



t ← y ;
y ← x;
x ← z;
z ← t ; *{t contient l'ancienne valeur de y}*

Algorithme

```
Algorithme Exo2_2;
  Variables
       x, y, z , t : entier;
Début
    {Entrées}
    Lire(x, y, z) ;
    {Traitement}
    t ← y;
    y ← x;
    x ← z ;
    z ← t ;
    {Sorties}
    Écrire(x, y, z) ;
Fin.
```

Programme C

```c
#include <stdio.h>

int main()
{
    int x, y, z, t;
    /* Entrées */
    printf("Donner deux valeurs entières X, Y et Z : ");
    scanf("%d %d %d", &x, &y, &z);

    /* Traitement */
   t = y;
   y = x;
   x = z;
   z = t;

    /* Sortie */
   printf("X = %d ,   Y = %d    et   Z = %d", x, y, z);

   return 0;
}
```

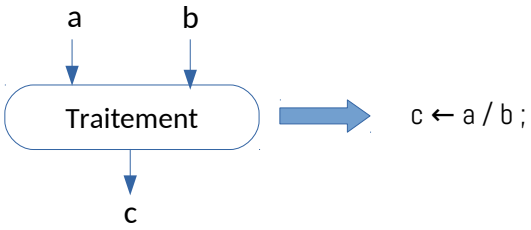
Voir le lien : https://onlinegdb.com/0LtqiwpOB

Dérouler l'algorithme (ou bien le programme) pour X=1, Y=20 et Z=33

Instructions	Variables			
	x	y	z	t
Lire (x, y, z)	1	20	33	/
t ← y	"	"	"	20
y ← x	"	1	"	"
x ← z	33	"	"	"
z ← t	"	"	20	"
Écrire (x,y, z)	33	1	20	"

3 – Calculer la division entre deux variables réelles a et b

Analyse du problème :



Algorithme

```
Algorithme Exo2_3;
  Variables
      a, b, c : réel;
Début
    {Entrées}
    Lire(a, b) ;

    {Traitement}
    c ← a / b;

    {Sorties}
    Écrire(b) ;
Fin.
```

Programme C

```c
#include <stdio.h>

int main()
{
    float a, b, c;
    /* Entrées */
    printf("Donner a et b : ");
    scanf("%f %f", &a, &b);

    /* Traitement */
   c = a/b;

    /* Sortie */
   printf("a / b =  %f", c);

   return 0;
}
```

Voir le lien : https://onlinegdb.com/08ywvZlsr

Dérouler l'algorithme pour a=10 et b=2.5

Instructions	Variables		
	a	b	c
Lire (a, b);	10	2.5	/
c ← a / b = 10 / 2.5 = 4	"	"	4
Écrire (c);	"	"	4

Remarque :

L'algorithme précédent n'est pas complet, puisque la valeur de b n'a pas été vérifiée : la valeur de b doit être différente de 0 pour pouvoir réaliser la division a / b.

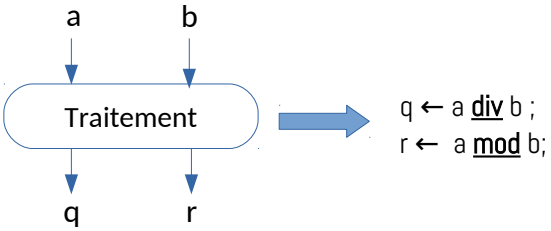
4 – Calculer la division euclidienne entre deux entiers a et b

Analyse du problème :

La division euclidienne entre deux entiers a et b est écrite écrite comme suit :

a = b * q + r, tel-que q est quotient et r et le reste de division.

Donc, nous aurons le schéma d'entrée, sortie et traitement ci à droite :



Algorithme

```
Algorithme Exo2_4;
  Variables
       a, b, q, r : entier;
Début
    {Entrées}  Lire(a, b) ;
    {Traitement}
    q ← a div b;
    r ← a mod b;
    {Sorties}
    Écrire(q, r) ;
Fin.
```

Programme C

```c
#include <stdio.h>

int main()
{
    int a, b, q, r;
    /* Entrées */
    printf("Donner a et b : ");
    scanf("%d %d", &a, &b);

    /* Traitement */
   q = a / b;
   r = a % b;

    /* Sortie */
   printf("Q = %d et R = %d", q, r);

   return 0;
}
```

Voir le lien : https://onlinegdb.com/AcmDsUOjh

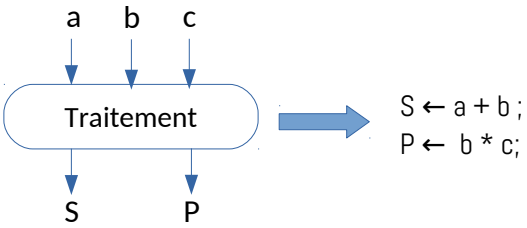
5 – Calculer la somme de a et b et le produit de b et c

Analyse du problème :

La somme de a et b sera enregistré dans la variable S, donc on écrit : S ← a + b;

Le produit de b et c sera stocké dans la variable P, on écrit alors : P ← b * c ;

Donc, nous aurons le schéma d'entrée, sortie et traitement ci à droite:



Algorithme

```
Algorithme Exo2_5;
  Variables
       a, b, c S, P : entier;
Début
    {Entrées}  Lire(a, b, c) ;
    {Traitement}
    S ← a + b;
    P ← b * c;
    {Sorties}
    Écrire(S, P) ;
Fin.
```

Programme Pascal

```c
#include <stdio.h>
Int main() {
    int a, b, c, S, P;
    printf("Donner a, b et c : ");
    scanf("%d %d %d", &a, &b, &c);

    S = a + b;    P = b * c;

    printf("a+b = %d et b*c=%d", S, P);
    return 0;
}
```

Link : https://onlinegdb.com/L8ROtnqKyi

6– Calculer la valeur absolue, le carré et la racine carrée d'un nombre

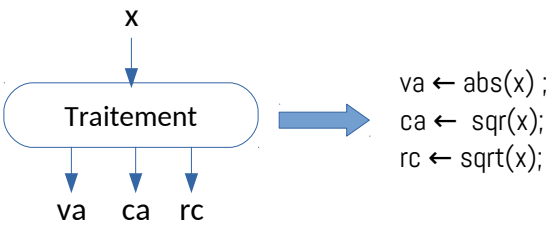
Analyse du problème :

Soit x un nombre réel, sa valeur absolue va sera obtenue par l'affectation : va ← abs(x) ;

Son carré, ca, sera calculé par : ca ← sqr(x) ;

Sa racine carrée, rc, est obtenue par l'affectation : rc ← sqrt(x);

Ainsi, nous obtenons le schéma d'entrée, sortie et traitement ci à droite:



Algorithme

```
Algorithme Exo2_6;
  Variables
       x, va, ca, rc : réel;
Début
    {Entrées}
    Lire(x) ;

    {Traitement}
    va ← abs(x);
    ca ← sqr(x);
    rc ← sqrt(x);

    {Sorties}
    Écrire(va, ca, rc) ;
Fin.
```

Programme C

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int main()
{
    /* Déclarations */
    float x, va, ca, rc;

    /* Entrées */
    printf("Donner une valeur réelle X : ");
    scanf("%f", &x);

    /* Traitement */
    va = abs(x);
    ca = pow(x, 2);
    rc = sqrt(va);

    /* Sortie */
    printf("La valeur absolue de X : %.3f\n", va);
    printf("Le carrée de X : %.3f\n", ca);
    printf("La racine carrée de X : %.3f\n", rc);
    return 0;
}
```

Voir le lien : https://onlinegdb.com/2gdDwOusf

Bon Courage
&
Travaillez bien.

Cours Elearning :
https://elearning.univ-bejaia.dz/course/view.php?id=7944

Page facebook :
https://www.facebook.com/InitiationAlgoProgrammation/

La chaîne Youtube :
https://www.youtube.com/c/AlgoProgrammation1èreAnnéeTechnologie

La playlist sur le langage C :
https://youtube.com/playlist?list=PLwHHAvorm5F-tL9EXDEH0miOKmAj7iUTU

Adapté par: Redouane OUZEGGANE
rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz