



Pr. Youness KHOUREDIFI, PhD en Informatique
Professeur à la Faculté Polydisciplinaire – Khouribga –
Université Sultan Moulay Slimane – Béni Mellal –
Consultant IT : SQL 2016 Database Administration, Core
Infrastructure 2016, Azure Solutions Architect Expert,
Data Analyst Associate, Ingénieur DevOps.
y.khourdifi@usms.ma

TECHNIQUES DE PROGRAMMATION EN C



Chapitre

Les techniques de programmation en C

III. 1 . Introduction au langage C :

3

Bref historique

- ❑ Le langage C a été créé par Dennis Ritchie aux ‘Bell Telephone Laboratories’ en 1972: il a été conçu pour développer le système d’exploitation (UNIX).
- ❑ En raison de sa puissance et de sa souplesse, le langage C s’est rapidement répandue au-delà des laboratoires Bell.
- ❑ Une première définition de ce langage est apparue en 1978 avec l’ouvrage de Kernighan et Ritchie *The C programming language*
- ❑ Les programmeurs ont commencé à l’utiliser pour écrire toutes sortes de programmes.

Dennis Ritchie
1941-2011



III. 1 . Introduction au langage C :

4

Bref historique



L'implémentation est devenu un casse-tête pour les programmeurs.

Diverses organisations ont utilisé leurs propres versions du langage C

Son succès international

Normalisation, par l'ANSI en 1983, et l'ISO, et en 1993 par le CEN et enfin, en 1994, par l'AFNOR

- ANSI (American National Standard Institute)
- ISO (International Standards Organization)
- CEN (Comité européen de normalisation)
- AFNOR (Association française de normalisation).

III. 1 . Introduction au langage C :

5

Pourquoi utiliser le langage C

- ❑ **Souple et puissant** : Le langage C est utilisé pour des projets variés comme les systèmes d'exploitation, les traitements de textes, les graphiques, les tableurs, les compilateurs pour d'autres langages.
- ❑ Il existe un large choix de compilateurs et d'utilitaires.
- ❑ **Le langage C est modulaire**: son code peut(et devrait) être écrit sous forme de sous-programme appelés fonctions. Si vous passez des informations à ces fonctions, vous obtenez un code réutilisable.
- ❑ C est un langage structuré (offre plusieurs structures de contrôle) et typé (déclarations obligatoires).

III. 1 . Introduction au langage C :

6

Cycle de développement du programme :

Compilation du code source :

- ❑ Votre ordinateur ne peut pas comprendre le code source. Il ne peut comprendre que des instructions binaires dans ce que l'on appelle du **langage machine**. Votre programme C doit être transformé en langage machine pour pouvoir être exécuté sur votre ordinateur. Cette opération est réalisée par un **compilateur** qui transforme votre fichier code source en un fichier contenant le code objet (les mêmes instructions en langage machine).
- ❑ Chaque compilateur possède sa propre commande pour créer le code objet.
 - ❑ Windows: C Microsoft, Turbo C, Turbo C++, Borland C, Borland C++, code::blocks.
 - ❑ Unix : utiliser la commande c.
 - ❑ Linux et Unix: la commande gcc.

III. 1 . Introduction au langage C :

7

Cycle de développement du programme :

Création du fichier exécutable :

- ❑ Une partie du langage C est constituée d'une bibliothèque de fonctions contenant du code objet destiné à des fonctions prédéfinies.
- ❑ Ces fonctions sont fournies avec votre compilateur. Si votre programme les utilise, le fichier objet obtenu après compilation doit être complété par le code objet issu de la bibliothèque de fonctions. Cette dernière étape, appelée liaison, fournit le programme exécutable (exécutable signifie que ce programme peut être exécuté sur votre ordinateur)

III. 1 . Introduction au langage C :

8

Cycle de développement du programme :

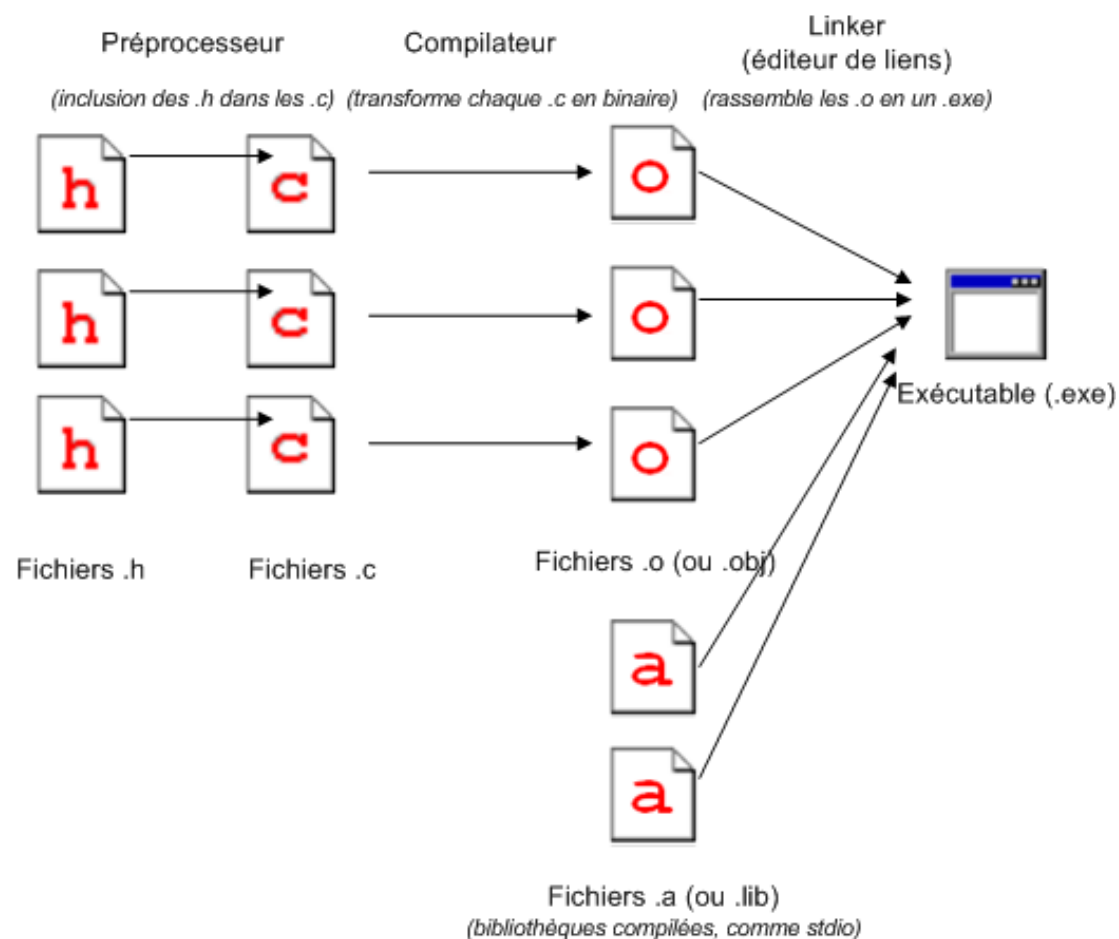
Processus de construction d'un exécutable :

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7  }
```



Hello world!

Process returned 0 (0x0) execution time : 0.016 s
Press any key to continue.



III. 1 . Introduction au langage C :

9

Exemple de programme en langage C : Exemple_01

```
1  #include <stdio.h>
2  #include <math.h>
3  #define NFOIS 5
4  main()
5  { int i ;
6    float x ;
7    float racx ;
8    printf ("Bonjour\n") ;
9    printf ("Je vais vous calculer %d racines carrées\n", NFOIS) ;
10   for (i=0 ; i<NFOIS ; i++)
11   { printf ("Donnez un nombre : ") ;
12     scanf ("%f", &x) ;
13     if (x < 0.0)
14     printf ("Le nombre %f ne possède pas de racine carrée\n", x) ;
15     else
16     { racx = sqrt (x) ;
17       printf ("Le nombre %f a pour racine carrée : %f\n", x, racx) ;
18     }
19   }
20   printf ("Travail terminé - Au revoir") ;
21 }
```

Bloc

Programme principal

```
Bonjour
Je vais vous calculer 5 racines carrees
Donnez un nombre : 7
Le nombre 7.000000 a pour racine carree : 2.645751
Donnez un nombre : 16
Le nombre 16.000000 a pour racine carree : 4.000000
Donnez un nombre : 8.23
Le nombre 8.230000 a pour racine carree : 2.868798
Donnez un nombre : -7
Le nombre -7.000000 ne possède pas de racine carree
Donnez un nombre : 25
Le nombre 25.000000 a pour racine carree : 5.000000
Travail termine - Au revoir
Process returned 0 (0x0)   execution time : 25.588 s
Press any key to continue.
```



Un bloc peut lui-même contenir d'autres blocs. En revanche, nous verrons qu'une fonction ne peut jamais contenir d'autres fonctions.

III. 1 . Introduction au langage C :

10

Exemple de programme en langage C : Exemple_02

```
1  #include <stdio.h>
2  main()
3  {
4      char op ;
5      int n1, n2 ;
6      printf ("opération souhaitée (+ ou *) ? ") ;
7      scanf ("%c", &op) ;
8      printf ("donnez 2 nombres entiers : ") ;
9      scanf ("%d %d", &n1, &n2) ;
10     if (op == '+')
11         printf ("leur somme est : %d ", n1+n2) ;
12     else
13         printf ("leur produit est : %d ", n1*n2) ;
14 }
```

```
opération souhaitée (+ ou *) ? +
donnez 2 nombres entiers : 63 7
leur somme est : 70
Process returned 0 (0x0)   execution time : 7.199 s
Press any key to continue.
```

III. 1 . Introduction au langage C :

11

Les délimiteurs: Ils constituent la ponctuation du langage :

Le délimiteur	Utilité
Les accolades { }	Elle contiennent le corps du programme.
Le point virgule ;	La fin d'une ligne est marquée par un ; et nom par un retour à la ligne. Donc sur une même ligne de l'écran on peut avoir plusieurs lignes du programme. Ces lignes se terminent par un point virgule.
l'espace blanc	il sert à séparer deux identificateurs. exemple : int a ;
Le signe =	pour affecter la valeur d'une expression ou bien pour affecter une valeur à une variable identifiée par son nom. Exemple A= 4 ; B= A*2 ;
La virgule ,	elle sépare les variables dans l'étape de déclaration.
Le signe ++	pour augmenter d'une unité la valeur d'une variable Exemple : A=4 ; B=A ; A++, B= B+A ; Le résultat pour B est 9

III. 1 . Introduction au langage C :

12

Les délimiteurs: Ils constituent la ponctuation du langage :

Le délimiteur	Utilité
Le signe --	<p>pour diminuer d'une unité la valeur d'une variable</p> <p>Exemple : A=4 ;</p> <p style="padding-left: 40px;">B=A ;</p> <p style="padding-left: 40px;">A--,</p> <p style="padding-left: 40px;">B= B+A ;</p> <p>Le résultat pour B est 7</p>
Le signe ()	<p>vient juste avant l'accolade qui marque le début du programme.</p>
Le signe " "	<p>pour délimiter les messages que l'on veut afficher à l'écran lors de l'exécution du programme.</p>
Le signe _	<p>permet de séparer les mots dans un identificateur.</p> <p>Exemple : premiere_valeur=2 ;</p> <p style="padding-left: 40px;">Deuxieme_valeur=3 ;</p>

III. 1 . Introduction au langage C :

13

Quelques règles d'écriture :

Les identificateurs :

- ❑ Les identificateurs en langage C comme dans la plupart des langages, ils sont formés d'une suite de caractères choisis parmi les lettres ou les chiffres, le premier d'entre eux étant nécessairement une lettre.
- ❑ Le caractère souligné (`_`) est considéré comme une lettre. Il peut donc apparaître au début d'un identificateur.
- ❑ Les majuscules et les minuscules sont autorisées mais ne sont pas équivalentes. Ainsi, en C, les identificateurs `ligne` et `Ligne` désignent deux objets différents.
- ❑ Par contre on peut ajouter autant des espaces dans le code



En ce qui concerne la longueur des identificateurs, la norme ANSI prévoit qu'au moins **les 31 premiers caractères** soient « significatifs » (autrement dit, deux identificateurs qui diffèrent par leurs 31 premières lettres désigneront deux objets différents).

III. 1 . Introduction au langage C :

14

Quelques règles d'écriture :

Les mots-clés :

- ❑ Certains « mots-clés » sont réservés par le langage à un usage bien défini et ne peuvent pas être utilisés comme identificateurs. En voici la liste, classée par ordre alphabétique.

Les mots-clés du langage C

auto	default	float	register	struct	volatile
break	do	for	return	switch	while
case	double	goto	short	typedef	
char	else	if	signed	union	
const	enum	int	sizeof	unsigned	
continue	extern	long	static	void	

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

15

Les fonctions d'entrée et sortie (printf() et scanf()) :

□ Récupérer une saisie :

- On va demander à l'utilisateur de taper un nombre dans la console.
- Ce nombre, on va le récupérer et le stocker dans une variable.
- Pour demander à l'utilisateur d'entrer quelque chose dans la console, on va utiliser une autre fonction toute prête : `scanf`.
- D'une manière générale, l'appel de `scanf` se présente ainsi :
 - `scanf ("format", liste d'adresses)`

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

16

Les fonctions d'entrée et sortie (printf() et scanf()) :

❑ Récupérer une saisie :

❑ Exemple:

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

int main ()    // Équivalent de int main(int argc, char *argv[])
{
    int age = 0; // On initialise la variable à 0
    printf ("Quel age avez-vous ? "); // On demande d'entrer l'âge avec printf
    scanf ("%d", &age);
    printf ("Ah ! Vous avez donc %d ans !\n\n", age);
    return 0;
}
```


II. 2 . Représentation des éléments de base d'un algorithme en langage C :

17

Les fonctions d'entrée et sortie (printf() et scanf()) :

□ Afficher le contenu d'une variable :

On utilise `printf` de la même manière, sauf que l'on rajoute un symbole spécial à l'endroit où l'on veut afficher la valeur de la variable.

L'appel à de `printf` se présente comme ceci :

```
printf ("format", liste_d'expressions )
```

Par exemple :

```
printf ("Votre note est %d" , note);
```

Ce « symbole spécial » est en fait un '%' suivi d'une lettre. Cette lettre permet d'indiquer ce que l'on doit afficher. 'd' signifie que l'on souhaite afficher un int.

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

18

Les fonctions d'entrée et sortie (printf() et scanf()) :

□ Afficher le contenu d'une variable :

Format	Type attendu
%d	entier décimal
%f	réel
%e	Réel flottant
%c	caractère (1 seul)
%s	chaîne de caractères



On peut afficher la valeur de plusieurs variables dans un seul `printf`. Il vous suffit d'indiquer des `%d` ou des `%f` là où vous voulez, puis d'indiquer les variables correspondantes dans le même ordre, séparées par des virgules.

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

19

Les opérateurs et les expressions :

L'opération par laquelle on attribue une valeur à une variable s'appelle affectation. Dans un algorithme (pseudo-code) " \leftarrow ".

Dans le langage C par le signe " = ".

```
#include <stdio.h>
main()
{
    int i;
    i = i*2+5;
    printf ("b= %d", i);
}
```

i : variable (Adresse, type , valeur)

$i*2+5$: expression (type, valeur)

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

20

Les opérateurs Arithmétique, de comparaison & logique :

Arithmétique	
Opérateur	C
Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo (reste de la division entière)	%

Comparaison	
Opérateur	C
Supérieur stricte	>
Inférieur stricte	<
Supérieur ou égal	>=
Inférieur ou égal	<=
Egal	==
Différent	!=

Logique	
Opérateur	C
ET Logique	&&
OU Logique	
Négation Logique	!

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

21

Les Opérateurs d'incrémentations & de décrémentation :

++ Incrément

-- Décrément

On dit que ++ est :

- ❑ Un opérateur de **pré incrémentation** lorsqu'il est placé à gauche de la variable
- ❑ Un opérateur de **post incrémentation** lorsqu'il est placé à droite de la variable

a++;	a = a+1;
b = a++;	b = a; a = a+1;
b = ++a;	a = a+1; b = a;

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

22

Les Opérateurs d'incrémentations & de décrémentation :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a, b, c;
    a = 5;
    b = a + 4;
    b = a++ - 7;
    b = a - b - 7;
    printf("a= %d\n", a);
    printf("b= %d\n", b);
    system("pause");
    return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a, b, c;
    a = 5;
    b = a + 4;
    b = ++a - 7;
    b = a - b - 7;
    printf("a= %d\n", a);
    printf("b= %d\n", b);
    system("pause");
    return(0);
}
```

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

23

Les opérateurs d'affectation élargie :

❑ C dispose d'opérateurs encore plus puissants.

$i += k$ ($i = i + k$)

$i -= k$ ($i = i - k$)

$a *= b$ ($a = a * b$)

$a /= b$ ($a = a / b$)

$a \% = b$ ($a = a \% b$)

```
#include <stdio.h>
main()
{
    int a, b, c;
    a = 5;
    b = 2;
    a += 4;
    b *= a;
    b = a++ - 7;
    b += a - 7;
    a %= b;
    printf("a= %d\n", a);
    printf("b= %d\n", b);
    system("pause");
}
```

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

24

Exercice :

□ Ecrire un programme en C qui lit trois variables réelles A, B et C et qui calcule :

$$X = A + B$$

$$Y = (A * B) - C$$

$$W = A + C$$

$$Z = (A - B) / (A + C)$$

On utilisera les commentaires autant que possible

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

25

Solution de l'Exercice :

```
//Un premier programme C
#include <stdio.h>
#include <stdlib.h>
//Déclaration
int main()
{
    float A,B,C,X,Y,Z,W;
    //lecture des données
    printf("donner trois reels A,B et C ");
    scanf("%f%f%f", &A,&B,&C);

    //traitement
    X=A+B;    /* calcul de X*/
    Y=(A*B)-C; /* calcul de Y */
    W=A+C;    /* calcul de W */
    Z=(A-B)/(A+C); /* si W est non nul on calcule puis on affiche Z */
    //Edition des résultats
    printf("Z= %4.2f\n", Z);
    printf("FIN");
}
```

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

26

Structure d'un programme C :

Un programme écrit en langage C est composé de deux parties :

□ Partie 1 : Les déclarations

Elle comporte la déclaration des fonctions des bibliothèques (bibliothèque standard ou autre) par inclusion des fichiers fournis avec le langage et peut comprendre des déclarations des variables « globales ».

La première instruction **#include <stdio.h>** est une directive au compilateur dont le but est de réaliser l'inclusion mentionnée : ici, l'inclusion des fonctions d'entrée/sortie intégrées à la bibliothèque standard.

On peut également faire appel à d'autres bibliothèques, par exemple une bibliothèque mathématique. Dans ce cas, l'inclusion dans le code par **#include <math.h>** devra s'accompagner d'une prise en compte explicite de la bibliothèque à la compilation.

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

27

Structure d'un programme C :

□ Partie 1 : Les déclarations :

- Les Inclusion de source `#include<nonfichier.h>`

Nom fichier	Rôle
<code><math.h></code>	pour utiliser les fonctions mathématiques usuelles
<code><stdio.h></code>	pour utiliser les fonctions d'entrées/sortie « printf() »
<code><stdlib.h></code>	Pour la gestion de la mémoire, conversions et les fonctions système
<code><string.h></code>	Pour manipuler des chaines de caractère.
<code><conio.h></code>	Pour utiliser les fonctions de contrôle l'affichage à l'écran : « getch() »
<code><complex.h></code>	Pour manipuler les nombres complexes
<code><ctype.h></code>	pour utiliser les fonctions de manipulation des caractères (fonction test) : « islower() » : test si un caractère est majuscule.

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

28

Structure d'un programme C :

□ Partie 2 : Le corps du programme :

- Tout programme C doit comporter une fonction principale **main**. Cette fonction est celle utilisée par le système pour exécuter le programme.
- **Instructions** : constituent le corps du programme. Elles sont plus ou moins complexes et nombreuses selon les programmes.
- La partie **Les déclarations** peut contenir des définitions de fonctions qui seront utilisées par l'intermédiaire de la fonction principale **main**.

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

29

Structure d'un programme C :

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

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

} Directives de préprocesseur

} Instructions

} Fonction

Structure d'un programme en langage C

```
#include <stdio.h>

int main
{
    ...
    return 0;
}
```

} Partie 1
déclaration

} Partie 2
Corps du programme

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

30

Structure d'un programme C :

- La fonction `scanf` est une fonction de saisie qui fonctionne avec un format décrivant la nature de l'information à lire et donc la conversion à effectuer.
- La fonction `scanf` s'utilise de la manière suivante :
 - ▣ `scanf("%c", &un_caractere);`
 - ▣ `scanf("%d", &un_entier);`
 - ▣ `scanf("%f", &un_reel);`
 - ▣ `scanf("%lf", &un_double);`
 - ▣ `scanf("%c%d%lf", &un_caractere, &un_entier, &un_double);`

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

31

Structure d'un programme C :

- La fonction `printf` est une fonction de saisie qui, comme `scanf`, fonctionne avec un format décrivant la nature de l'information à écrire et donc la conversion à effectuer.
- La fonction `printf` s'utilise de la manière suivante :
 - ▣ `printf("%c\n", un_caractere);`
 - ▣ `printf("%d\n", un_entier);`
 - ▣ `printf("%f\n", un_reel);`
 - ▣ `printf("%f\n", un_double);`
 - ▣ `printf("2 * %d = %d\n", un_entier, un_entier * 2);`
 - ▣ `printf("c = %c et nb = %f\n", un_caractère, un_double);`
 - ▣ `printf("texte = %s \n", une_chaine_de_caractères);`

II. 2 . Représentation des éléments de base d'un algorithme en langage C :

32

Structure d'un programme C :

❑ On peut donc retenir le canevas suivant pour l'écriture d'un programme en C :

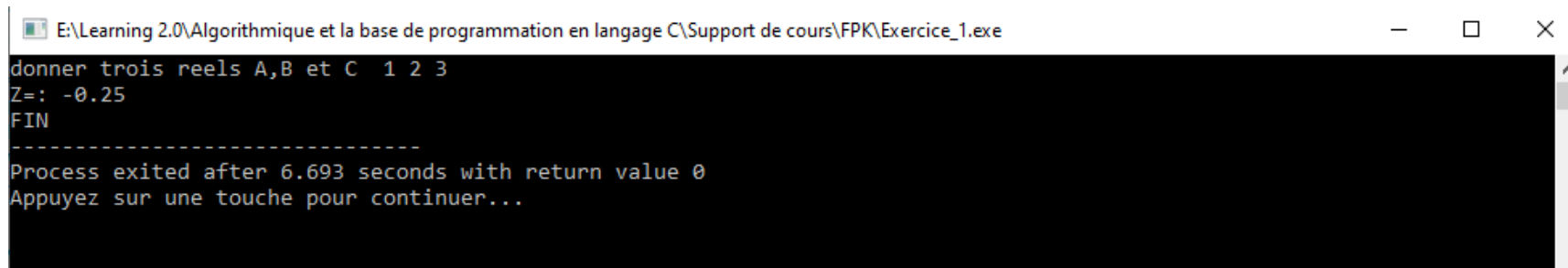
```
/******  
présentation du programme et des auteurs ainsi que la date  
*****/  
  
#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
  Déclaration des variables, constante ....etc ;  
  //lecture des données  
  //traitement  
  //Edition des résultats  
}
```


II. 2 . Représentation des éléments de base d'un algorithme en langage C :

33

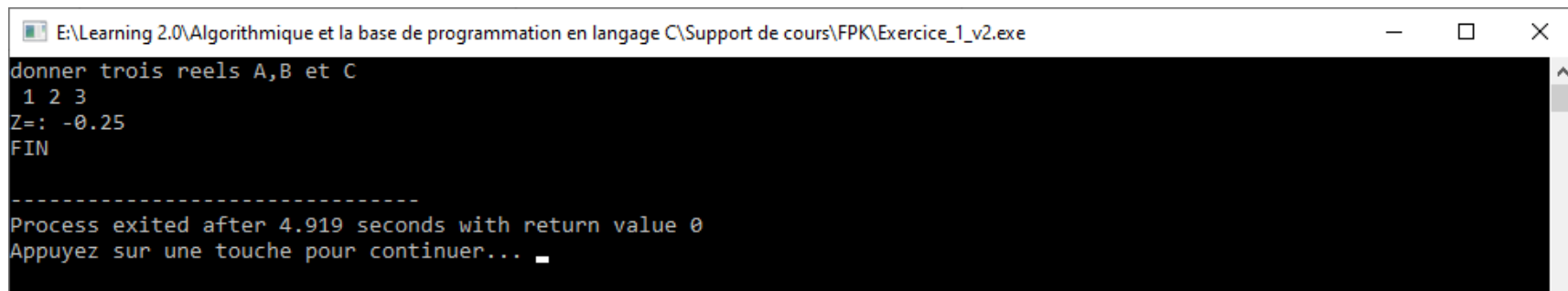
Exécution :

- ❑ L'écran d'exécution pour l'exercice précédent est par exemple:



```
E:\Learning 2.0\Algorithmique et la base de programmation en langage C\Support de cours\FPK\Exercice_1.exe
donner trois reels A,B et C 1 2 3
Z=: -0.25
FIN
-----
Process exited after 6.693 seconds with return value 0
Appuyez sur une touche pour continuer...
```

- ❑ On peut améliorer l'affichage sur cet écran d'exécution en ajoutant des lignes blanches de la manière suivantes :



```
E:\Learning 2.0\Algorithmique et la base de programmation en langage C\Support de cours\FPK\Exercice_1_v2.exe
donner trois reels A,B et C
1 2 3
Z=: -0.25
FIN
-----
Process exited after 4.919 seconds with return value 0
Appuyez sur une touche pour continuer...

```

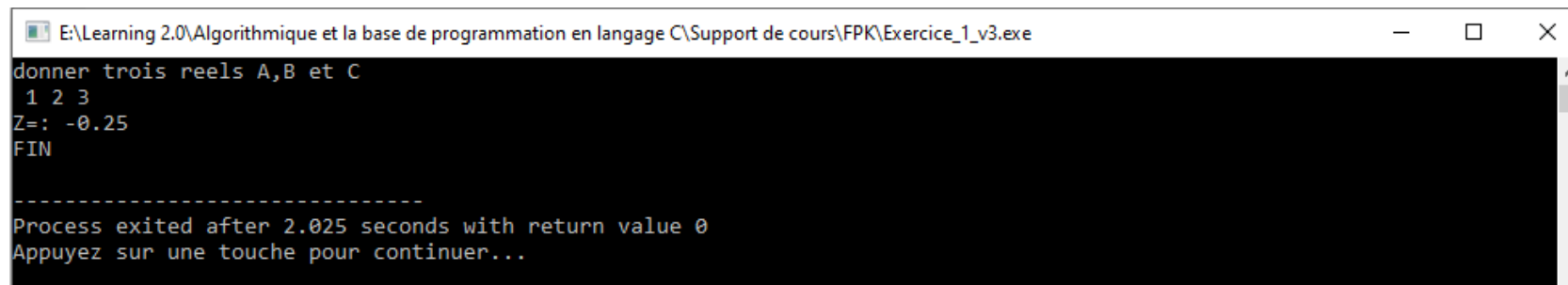
II. 2 . Représentation des éléments de base d'un algorithme en langage C :

34

Exécution :

❑ On peut aussi améliorer le programme afin qu'il ne calcule Z que si W est non nul, soit :

▪ **Exécution 1** : cas où W est non nul



```
E:\Learning 2.0\Algorithmique et la base de programmation en langage C\Support de cours\FPK\Exercice_1_v3.exe
donner trois reels A,B et C
1 2 3
Z=: -0.25
FIN

-----
Process exited after 2.025 seconds with return value 0
Appuyez sur une touche pour continuer...
```

▪ **Exécution 2** : cas où W est nul



```
E:\Learning 2.0\Algorithmique et la base de programmation en langage C\Support de cours\FPK\Exercice_1_v3.exe
donner trois reels A,B et C
1
2
-1
Votre denominateur est nul
FIN

-----
Process exited after 9.993 seconds with return value 0
Appuyez sur une touche pour continuer...
```

Chapitre

La sélection en algorithmique et en langage C

III. 1. La sélection en algorithmique :

36

- Dans cette structure, un certain nombre d'instructions est exécuté si la condition du test est vérifiée, sinon, c'est un autre groupe d'instructions qui est exécuté.
- On distingue quatre variantes de cette structure.

□ Sélection avec une seule alternative (structure alternative simple (SAS)) :

Lorsqu'il y'a une seule condition à vérifier, la syntaxe algorithmique se présente de la manière suivante :

Si (condition) { traitement }

III. 1. La sélection en algorithmique :

37



Qu'est ce qu'une condition ?

Une condition est une comparaison entre valeurs de même type

Elle signifie qu'une condition est composée de trois éléments :

- ❑ Une valeur;
- ❑ Un opérateur de comparaison;
- ❑ Une autre valeur,

❑ Exemple 1:

Calculer l'inverse d'un nombre

III. 1. La sélection en algorithmique :

38

❑ Exemple 1:

```
//Titre : calcul de l'inverse d'un nombre
//réalisé par Y. Khourdifi
//05 Novembre 2021
//déclaration
Nombre, invnombre :réel
// préparation du traitement
Lis (Nombre)
// traitement
Si (Nombre !=0 )
{ invnombre = 1/Nombre
Ecris (" inverse= ",invnombre)
}
// Edition des résultats
Ecris (" FIN ")
```

III. 1. La sélection en algorithmique :

39

❑ Sélection avec deux alternatives :

Lorsqu'il y'a deux conditions à vérifier, la syntaxe algorithmique peut se présenter deux manières :

❑ Première variante (structure alternative simple répétée (SASR))

Si (Condition1) {traitement1 }

Si (Condition2) {traitement2 }

❑ Deuxième variante (structure alternative complète (SAC))

Si (Condition1) {traitement1 }

Sinon {traitement2 }

III. 1. La sélection en algorithmique :

40

□ Exemple 2:

Calculer la valeur absolue d'un nombre

□ Exemple 3:

Calculer l'inverse d'un nombre avec un message d'erreur si le nombre égale a 0.

III. 1. La sélection en algorithmique :

41

□ Exemple 2:

```
//Titre : calcul de la valeur absolue d'un nombre  
//réalisé par Y. Khourdifi  
//05 Novembre 2021  
  
Nombre,absnombre :réel    //déclaration  
  
Ecris(" donnez un reel ") // préparation du traitement  
  
Lis (Nombre)  
  
Si (Nombre>=0) // traitement  
{ absnombre= Nombre  
Ecris (" val_absolue=  ",absnombre)  
}  
  
    Si (Nombre<0)  
{ absnombre= - Nombre  
Ecris (" val_absolue=  ",absnombre)  
}  
  
Ecris (" FIN ") // Edition des résultats
```

III. 1. La sélection en algorithmique :

42

□ Exemple 3:

```
//Titre : calcul de l'inverse d'un nombre  
//réalisé par Y. Khourdifi  
//05 Novembre 2021  
  
Nombre,invnombre :réel //déclaration  
  
Ecris " Donner nombre reel " // préparation du traitement  
  
Lis (Nombre)  
  
Si (Nombre != 0) // traitement  
{ invnombre = 1/ Nombre  
Ecris (" inverse = ",invnombre)  
}  
  
Sinon  
  
Ecris (" Votre nombre est nul, pas d'inverse ")  
Ecris (" FIN ") // Edition des résultats
```

III. 1. La sélection en algorithmique :

43

□ Exemple 4:

Editer le résultat pour la résolution d'une équation du second degré.

```
/* Titre : Résolution d'une équation du second degré  $A.X^2 + B.X + C = 0$   
avec A non nul */  
//réalisé par Y. Khourdifi  
//05 Novembre 2021
```

```
A,B,C,D :réel //déclaration  
// préparation du traitement  
Ecris " Donner trois reels A, B et C avec A non nul "  
Lis (A,B,C)  
D=B*B - 4*A*C // traitement  
Si (D>=0)  
{ Ecris (" solution(s) reelle ")}  
Sinon  
{Ecris (" solutions complexe ")}  
Ecris (" FIN ") // Edition des résultats
```

III. 1. La sélection en algorithmique :

44

❑ Sélection avec trois conditions ou plus :

❑ Première variante : (structure alternative simple répétée (S.A.S.R))

Cette structure peut aussi être utilisée lorsqu'on a plus de deux conditions.

Sa syntaxe est la suivante :

Si (Condition1) {traitement1 }

Si (Condition2) {traitement2 }

.

.

Si (ConditionN) {traitementN }

III. 1. La sélection en algorithmique :

45

□ Exemple 5:

Editer le résultat pour la résolution d'une équation du second avec A non nul.

```
/* Titre : Résolution d'une équation du second degré
A.X2 + B .X +C =0    avec A non nul    avec la S.A.S.R */
//réalisé par Y. Khourdifi
//05 Novembre 2021

A,B,C,D :réel //déclaration
// préparation du traitement
Ecris " Donner trois reels A, B et C avec A non nul "
Lis (A,B,C)
D=B*B - 4*A*C // traitement et édition
Si (D>0) { Ecris (" deux racines reelles ") }
Si (D<0) {Ecris (" deux racines complexes ")}
Si (D==0) {Ecris (" une racine double reelle ")}
    Ecris(" FIN de l algorithme ")
}
```

III. 1. La sélection en algorithmique :

46

❑ **Sélection avec trois conditions ou plus :**

❑ **Deuxième variante : Imbrication de la sélection (la structure alternative complète imbriquée (SACI))**

Elle est utilisée lorsqu'il y'a N conditions à vérifier (3 ou plus).

Sa syntaxe est la suivante :

```
Si (condition1) { traitement1}
    Sinon Si (condition2)      { traitement2}
    Sinon Si (condition3)      { traitement3}
    ...
    Sinon Si (conditionN-1)    { traitementN-1}
    Sinon                      { traitementN}
```

III. 1. La sélection en algorithmique :

47

□ Exemple 6:

Editer le résultat pour la résolution d'une équation du second degré uniquement sous forme de message en utilisant la structure alternative complète imbriquée.

```
/* Titre : Résolution d'une équation du second degré
A.X2 + B .X +C =0    avec A non nul avec la S.A.C.I */
//réalisé par Y. Khourdifi
//05 Novembre 2021

A,B,C,D :réel //déclaration
// préparation du traitement
Ecris " Donner trois reels A, B et C avec A non nul "
Lis (A,B,C)
D=B*B - 4*A*C // traitement et édition
Si (D>0)
{ Ecris (" deux racines reelles ")}
Sinon
Si (D<0)
{Ecris (" deux racines complexes ")}
Sinon
{Ecris (" une racine double reelle ")}
Ecris (" FIN ") // Edition des résultats
}
```

III. 1. La sélection en algorithmique :

48

❑ Sélection avec trois conditions ou plus :

❑ Troisième variante : Le choix multiple (la structure alternative de cas (S.A de Cas))

Elle est utilisée lorsqu'il y'a N conditions à vérifier ($N \geq 3$).

Cette structure peut se présenter de la façon suivante :

Identificateur : entier

Si (condition1) identificateur=1 ;

Si (condition2) identificateur=2 ;

...

Si (conditionN-1) identificateur=N-1 ;

...

Si (conditionN) identificateur=N ;

III. 1. La sélection en algorithmique :

49

□ Sélection avec trois conditions ou plus :

Selon (identificateur)

{

case 1 : {traitement1 ;break ;}

case 2 : {traitement2 ;break ;}

.

.

case N-1 : {traitementN-1 ;break ;}

default: {traitementN ;break ;} (au lieu de default, on peut aussi écrire case N)

}

III. 1. La sélection en algorithmique :

50

□ Sélection avec trois conditions ou plus :

- Cette structure est aussi appelée la **structure du switch**.
- Le corps de l'instruction switch prend la forme d'un **bloc { }** renfermant une suite d'instructions.
- Le fonctionnement de cette instruction est le suivant :

Expression est évaluée ;

- { s'il existe un énoncé cas avec une constante qui égale la valeur de expression, le contrôle est transféré à l'instruction qui suit cet énoncé ;
- { si un tel cas n'existe pas, et si l'énoncé default existe, alors le contrôle est transféré à l'instruction qui suit l'énoncé default ;
- { si la valeur d'expression ne correspond à aucun cas et s'il n'y a pas d'énoncé default, alors aucune instruction n'est exécutée.

III. 1. La sélection en algorithmique :

51

□ Exemple 7:

Editer le résultat pour la résolution d'une équation du second avec A non nul, utiliser la structure alternative de cas.

III. 1. La sélection en algorithmique :

52

□ Exemple 7:

```
/* Titre : Résolution d'une équation du second degré
A.X2 + B .X +C =0    avec A non nul Avec la structure alternative de cas*/
//réalisé par Y. Khourdifi
//05 Novembre 2021
A,B,C,D :réel //déclaration
I :entier
// préparation du traitement
Ecris " Donner trois reels A, B et C avec A non nul "
Lis (A,B,C)
D=B*B - 4*A*C // traitement et édition
Si (D>0)  I=1
Sinon
Si (D<0) I=2
Sinon
I=3
selon(identificateur: I) // Edition des résultats
{
Cas 1:{ { Ecris (" deux racines  reelles ") ;break ;}
Cas 2:{ {Ecris (" deux racines complexes ");break ;}
Cas 3:{ {Ecris (" une racine double reelle ") ;break ;}
}
    Ecris(" FIN de l algorithme ")
}
```

III. 1. La sélection en algorithmique :

53

□ Sélection avec une seule alternative (structure alternative simple (SAS)):

Lorsqu'il y'a une seule condition à vérifier, la syntaxe en C se présente de la manière suivante.

```
if (condition) { traitement ;}
```

III. 1. La sélection en algorithmique :

54

□ Exemple 8:

Calculer l'inverse d'un nombre

```
/*Titre : calculer l'inverse d'un nombre en utilisant la selection avec une seule  
alternative (S.A.S) */  
//réalisé par Y. Khourdifi  
//05 Novembre 2021  
#include <stdio.h>  
#include <stdlib.h>  
int main()  
{  
float Nombre, invnombre; //Déclaration  
scanf("%f", &Nombre); // préparation du traitement  
if (Nombre !=0 ) // traitement  
{ invnombre= 1/ Nombre ;  
printf("inverse=   %4.2f\n", invnombre);  
}  
printf("FIN \n"); // Edition des résultats  
}
```

```
12  
inverse=   0.08  
FIN  
  
-----  
Process exited after 8.106 seconds with return value 0  
Appuyez sur une touche pour continuer... _
```

III. 1. La sélection en algorithmique :

55

❑ Sélection avec deux alternatives

Lorsqu'il y'a deux conditions à vérifier, la syntaxe algorithmique peut se présenter en deux manières :

❑ Première variante (structure alternative simple répétée (SASR))

```
If (Condition1) {traitement1 ;}
```

```
If (Condition2) {traitement2 ;}
```

❑ Deuxième variante (structure alternative complète (SAC))

```
if (Condition1) {traitement1 ; }
```

```
else {traitement2 ;}
```

III. 1. La sélection en algorithmique :

56

□ Exemple 9:

Calculer la valeur absolue d'un nombre

□ Exemple 10:

Calculer l'inverse d'un nombre avec un message d'erreur si le nombre égale a 0,

III. 1. La sélection en algorithmique :

57

□ Exemple 9:

```
donnez un reel  
-18  
val_absolue= 18.00  
FIN  
  
-----  
Process exited after 7.997 seconds with return value 0  
Appuyez sur une touche pour continuer...
```

```
donnez un reel  
15  
val_absolue= 15.00  
FIN  
  
-----  
Process exited after 2.506 seconds with return value 0  
Appuyez sur une touche pour continuer... _
```

III. 1. La sélection en algorithmique :

58

❑ **Exemple 9:** */*Titre : calculer la valeur absolue d'un nombre
en utilisant structure alternative simple répétée */
//réalisé par Y. Khourdifi
//05 Novembre 2021
#include <stdio.h>
#include <stdlib.h>
int main()
{
float Nombre,absnombre; //déclaration
printf("donnez un reel \n"); // préparation du traitement
scanf("%f", &Nombre);
if (Nombre>=0) // traitement
{ absnombre= Nombre;
printf("val_absolue= %4.2f\n", absnombre);
}
if (Nombre<0)
{ absnombre= - Nombre;
printf("val_absolue= %4.2f\n", absnombre);
}
printf("FIN \n"); // Edition des résultats
}*

III. 1. La sélection en algorithmique :

59

□ Exemple 10:

```
15
inverse=  0.07
FIN

-----
Process exited after 2.926 seconds with return value 0
Appuyez sur une touche pour continuer...
```

```
0
votre nombre est nul il n a pas d inverse
FIN

-----
Process exited after 2.006 seconds with return value 0
Appuyez sur une touche pour continuer...
```

III. 1. La sélection en algorithmique :

60

□ Exemple 10:

```
/*Titre : calculer l'inverse d'un nombre  
en utilisant la structure alternative complete */  
//réalisé par Y. Khourdifi  
//05 Novembre 2021  
#include <stdio.h>  
#include <stdlib.h>  
int main()  
{  
float Nombre, invnombre;           //Déclaration  
// préparation du traitement  
scanf("%f", &Nombre);  
if (Nombre !=0 )                   // traitement  
{ invnombre= 1/ Nombre ;  
printf("inverse=   %4.2f\n", invnombre);  
}  
if (Nombre ==0 )  
{ printf("votre nombre est nul il n a pas d inverse\n");  
}  
printf("FIN \n");                  // Edition des résultats  
}
```

III. 1. La sélection en algorithmique :

61

❑ Sélection avec trois conditions ou plus :

❑ Deuxième variante : Imbrication de la sélection (la structure alternative complète imbriquée (SACI))

Elle est utilisée lorsqu'il y'a N conditions à vérifier ($N \geq 3$).

```
if (condition1)           { traitement1 ;}
    else if (condition2)   { traitement2 ;}
    else if (condition3)   { traitement3;}
    .
    .
    else if (conditionN-1) { traitementN-1 ;}
        else               { traitementN;}

```

III. 1. La sélection en algorithmique :

62

□ Exemple 11:

Editer le résultat pour la résolution d'une équation du second degré uniquement sous forme de message en utilisant la structure alternative complète imbriquée.

III. 1. La sélection en algorithmique :

63

□ Exemple 11:

```
Donner 3 reels A,B et C avec A non nul
1 2 1
Une racine double réelle
FIN

-----
Process exited after 1.993 seconds with return value 0
Appuyez sur une touche pour continuer... █
```

```
Donner 3 reels A,B et C avec A non nul
1 0 -1
deux racines réelles
FIN

-----
Process exited after 11.48 seconds with return value 0
Appuyez sur une touche pour continuer... █
```

```
Donner 3 reels A,B et C avec A non nul
1 0 1
deux racines complexes
FIN

-----
Process exited after 2.11 seconds with return value 0
Appuyez sur une touche pour continuer... █
```

III. 1. La sélection en algorithmique :

64

□ Exemple 11:

```
/* Titre : Résolution d'une équation du second degré  $A.X^2 + B.X + C = 0$ 
avec A non nul avec la S.A.C.I */
//réalisé par Y. Khourdifi
//05 Novembre 2021
#include <stdio.h>
#include <stdlib.h>
int main()
{
float A,B,C,D; //déclaration
// préparation du traitement
printf(" Donner 3 reels A,B et C avec A non nul \n");
scanf("%f%f%f",&A,&B,&C);
D=B*B-4*A*C; // traitement
if(D>0) { printf(" deux racines reelles \n"); }
else
    if(D==0)
        { printf(" Une racine double reelle \n");}
    else
        { printf(" deux racines complexes \n");}
printf("FIN \n");
}
```


III. 1. La sélection en algorithmique :

65

□ Le choix multiple (la structure alternative de cas)

Elle est utilisée lorsqu'il y'a N conditions à vérifier($N \geq 3$).

Cette structure peut se présenter de la façon suivante :

```
case 1 :  
{  
  traitement1 ;  
  break ;  
}
```

```
int Identificateur ;  
if (condition1)  identificateur=1 ;  
if (condition2)  identificateur=2 ;  
.  
if (conditionN-1)  identificateur=N-1 ;  
if (conditionN)  identificateur=N ;  
  Switch (identificateur)  
  {  
    case 1 :{traitement1 ;break ;}  
    case 2 :{traitement2 ;break ;}  
    .  
    case N-1 :{traitementN-1 ;break ;}  
    default:{traitementN ;break ;}  
  }
```

III. 1. La sélection en algorithmique :

66

□ Exemple 12:

Editer le résultat pour la résolution d'une équation du second avec A non nul.

```
donner 3 reels A,B et C avec A non nul
1 2 1
Une racine double réelle
FIN

-----
Process exited after 3.202 seconds with return value 0
Appuyez sur une touche pour continuer... _
```

```
donner 3 reels A,B et C avec A non nul
1 0 -1
deux racines réelles
FIN

-----
Process exited after 2.39 seconds with return value 0
Appuyez sur une touche pour continuer... _
```

```
donner 3 reels A,B et C avec A non nul
1 0 1
deux racines complexes
FIN

-----
Process exited after 1.579 seconds with return value 0
Appuyez sur une touche pour continuer...
```