UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence

Algorithmique et structure de données 1

Chapitre 2:Algorithme séquentiel simple

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2024-2025

Plan du cours

1. **Exemples simples d'algorithmes de la vie courante**
2. **L'algorithme**
3. **L'organigramme**
4. **Le programme**
5. **Notion de variable**
6. **Les types de bases des variables**
7. **Instructions de base**
8. **Opérations arithmétiques**
9. **Exemples**
10. **Trace d'exécution d'un programme**

Exemples simples d'algorithmes de la vie courante

Avez-vous déjà réalisé une recette de cuisine ?

Avez-vous déjà suivi un mode d'emploi pour fabriquer une maquette de légo ?

Avez vous déjà suivi un traitement ?

Vous avez déjà exécuté des algorithmes.

Avez vous déjà créé une recette de cuisine ?

Avez-vous déjà montré le chemin à un touriste ?

Avez vous fait chercher un objet à quelqu'un par téléphone ?

Vous avez déjà fabriqué – et fait exécuter – des algorithmes.

Recette de cuisine

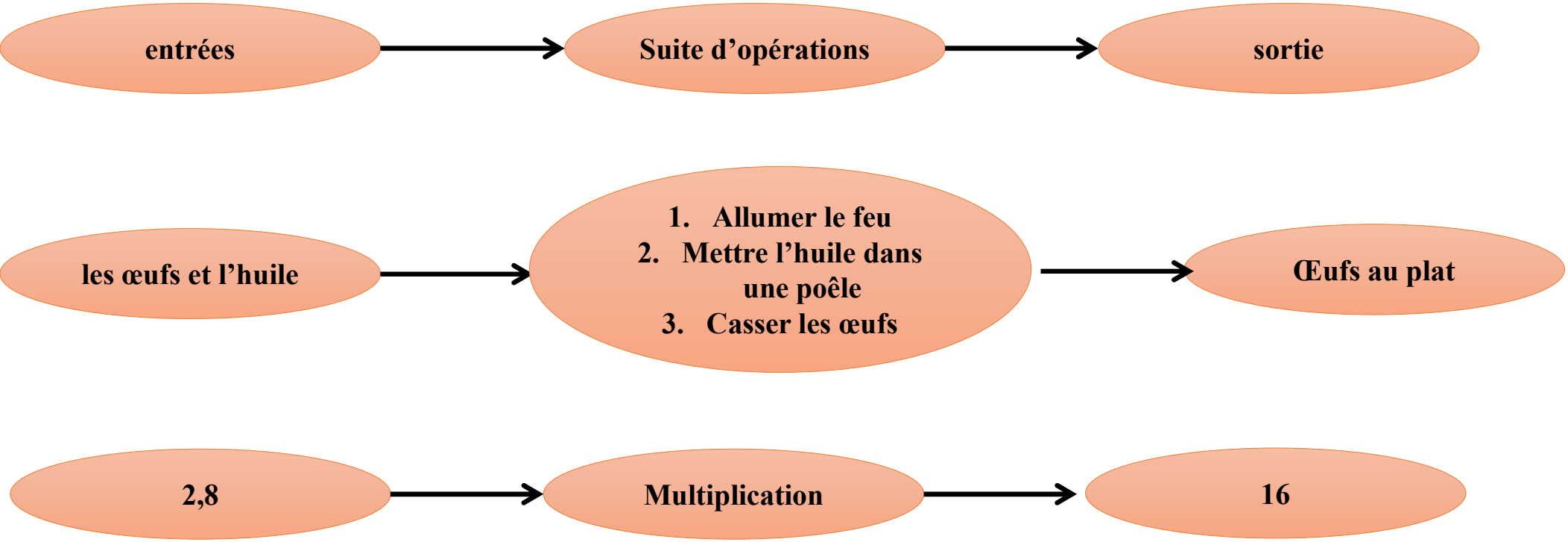
- Une recette de cuisine peut être vu comme un algorithme qui comporte trois étapes :
    1. Réunir les ingrédients
    2. Préparer
    3. Déguster (manger)
- La préparation d'un plat consiste à exécuter une suite d'instructions : par exemple, pour préparer un œuf au plat il faut casser les œufs et de l'huile avant de les mettre dans une poêle.
- En comparant avec les algorithmes de mathématiques, on pourrait dire que : les ingrédients de la recette sont les entrées du processus auxquelles on applique l'algorithme (la préparation) pour obtenir, en sortie, un plat que l'on déguste.

La calculatrice

Pour faire l'addition de deux nombres via la calculatrice, il faut:

1. **Allumer la Calculatrice.**
2. **Taper le 1er nombre.**
3. **Appuyer sur la touche (+)**
4. **Taper le 2eme nombre.**
5. **Appuyer sur la touche (=)**
6. **La calculatrice affiche le résultat.**

L’algorithme



L'algorithme

- **Le mot Algorithme vient du nom du mathématicien et astronome perse Muhammad ibn al-Khawarizmi, le père de l'algèbre, du « Abu Abdullah Muhammad ibn Musa al-Khwarizmi » qui a vécu au 9ème siècle.**
- **Un algorithme est une suite d'actions logiques et chronologiques qu'on doit suivre pour aboutir à la résolution d'un problème particulier.**
- **Ces actions sont constituées d'un nombre fini d'opérations élémentaires qui seront exécutées dans un ordre bien déterminé.**
- **Un algorithme est appelé aussi « pseudo-code »**

Structuration d'un algorithme

Algorithme nom_de_l'algorithme

Var nom_variable : type_variable;

Const nom_constante=valeur;

Début

Action 1 ;

Action 2 ;

……. <Partie Instructions>

Action n ,

Fin

L'en-tête

Les déclarations

Indentation

Le corps

Structuration d'un algorithme

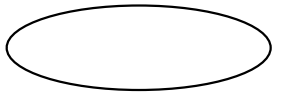
L'algorithme est composé de trois parties :

- **Entête : qui spécifie le nom de l'algorithme par exemple : Algorithme Somme ;**
- **Déclaration : contient la déclaration de différents objets (constantes, variables, ..) utilisées avec leurs types, ainsi que les modules (fonctions ou procédures) utilisés.**
- **Corps : Le corps d'un algorithme consiste en une suite d'opérations élémentaires (actions ou instructions), peuvent faisant appel à des fonctions ou procédures. Ces différentes opérations sont délimitées par les mots clefs Début et Fin.**

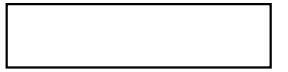
Remarque: Une constante est une information qui ne change de valeur dans la mémoire durant l'exécution de l'algorithme ou du programme

L'organigramme

L'organigramme est une représentation graphique de l'enchaînement d'une suite d'actions. Un organigramme utilise des symboles graphiques.

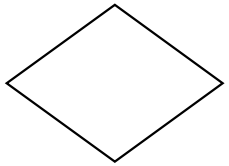


Le début et la fin d'un traitement



Les opérations de traitement

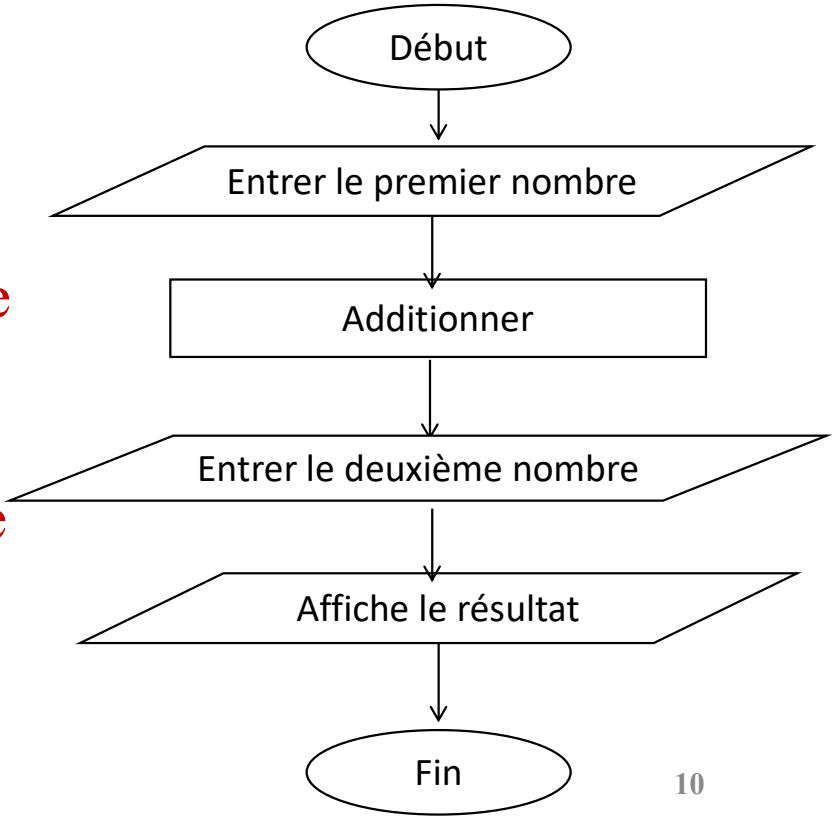
Une entrée ou une sortie



Test (condition)

Sens des actions

Exemple de l'addition avec la calculatrice



Programme

- **Un programme informatique est constitué d'une suite d'instructions ordonnées écrites dans un langage que l'ordinateur comprend.**
- **Une instruction informatique désigne une étape dans un programme informatique.**
- **Une instruction dicte à l'ordinateur l'action qu'il doit effectuer avant de passer à l'instruction suivante.**

Structuration d'un programme en C

```
# include <stdio .h>

# include <stdlib .h>

int main ()

{

// instruction

printf (" Salam ");

return 0;

}
```

Bibliothèques

Début du programme

Commentaire

Instruction qui permet d'afficher Salam

Fin du programme

Corps du programme

Structuration d'un programme en C

- # include permet d'ajouter des fichiers au projet , fichiers que l'on appelle bibliothèques.
  - stdio .h bibliothèque pour les fonctions d'entrées et de sorties.
  - stdlib .h bibliothèque pour la gestion de mémoire et autres.
- main () est appelée la fonction principale du programme, c'est par cette fonction que tous les **programmes** commencent.
- printf C'est une fonction prédéfinie qui permet l'affichage.
- Un commentaire ne change pas l'exécution du programme, il est uniquement là pour rendre le code plus lisible et compréhensible.
- Il existe deux types de commentaires; le commentaire sur une ligne qui est précédé par // ou le commentaire sous forme de paragraphe délimité par /* */.

Variables

Si on vous demande d'écrire un programme qui calcule la somme de deux entiers, a quoi pensez vous ????????????

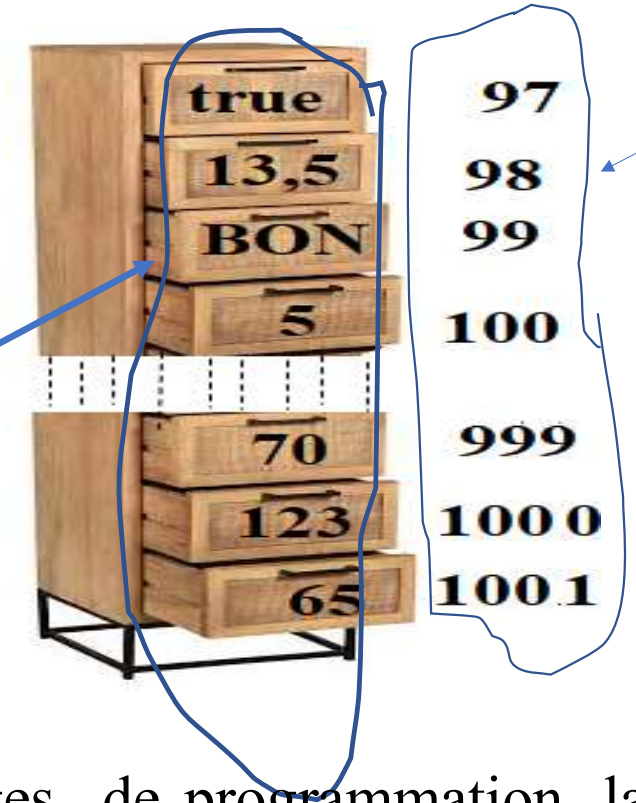
Problème de valeurs.

L'ordinateur doit pouvoir retenir les deux entiers avant de faire l'addition.

Il a besoin de mémoire.

Comment fonctionne la mémoire RAM

Les valeurs : à chaque adresse, on peut stocker une valeur (en réalité un nombre). On ne peut stocker qu'un nombre par adresse !



Les adresses : une adresse est un nombre qui permet à l'ordinateur de se repérer dans la mémoire vive.

Dans la plupart des langages de programmation, la taille de la mémoire d'une variable dépend du type de cette variable.

Notion de variables

- **Une variable est une information temporaire qu'on stocke dans la mémoire.**
- **On l'appelle ‘’variable” car c'est une valeur qui peut changer pendant le déroulement du programme.**
- **Une variable est caractérisée par:**
    1. **un nom : pour reconnaitre la variable.**
    2. **Un type: détermine l’ensemble des valeurs qu’elle peut prendre et les opérations réalisables qu’elle peut subir.**
    3. **une valeur : c'est le nombre qu'elle stocke, par exemple 7 ;**

Nommage d'une variable

Il y'a des contraintes et des conventions à respecter pour nommer une variable:

- Le nom d'une variable ne peut contenir que des minuscules, des majuscules et des chiffres (abcA012) ;
- Il doit commencer par une lettre ;
- Les caractères spéciaux sont interdits, on peut remplacer les espaces par le caractère Under score ''_'', C'est le seul caractère autorisé ;
- Les accents (éàê etc.) sont interdits.
- Il ne doit pas prendre le nom d'un mot réservé.

Exercice

Dans un tableau identifier les noms de variables valides et invalides : noteAlgo, 2025info, mail, nom, bac2025, numero_de_telephone, Prix-TTC, prénom, main.

Noms de variables valides	**Noms de variables invalides**
noteAlgo	2025info
mail	Prix-TTC
nom	prénom
bac2025	main
numero_de_telephone	

Bonnes pratiques pour nommer une variable

- Chaque programmeur a sa propre façon de nommer des variables;
- Commencer tous les noms de variables par une lettre minuscule;
- S'il y a plusieurs mots dans le nom de variable, mettez une lettre majuscule au début de chaque nouveau mot. Par exemple : NoteAlgo
- Faites en sorte de donner des noms clairs à vos variables. On aurait pu abréger NombreDePointGagnes, en l'écrivant par exemple ndpg.
- C'est peut-être plus court, mais c'est beaucoup moins clair pour vous quand vous relisez votre code. N'ayez donc pas peur de donner des noms un peu plus longs pour que ça reste compréhensible.

Les types d'une variable

- Lorsque vous créez une variable, vous devez indiquer son type.
- Les variables peuvent avoir cinq types de base dans un algorithme:

a) Type entier : un type numérique qui représente l'ensemble des entiers naturels et relatifs, tels que : 0, 45, -10,… Mot clé : entier (int ou long)

b) Type réel : un autre type numérique qui représente les nombres réels, tels que : 0.5, -3.67, 1.5e+5 ,… Mot clé : réel(float ou double)

Les types d'une variable

c) Type caractère : représente tous les caractères alphanumériques tels que : ’a’, ’B’, ’*’, ’9’, ’@’, ’ ’,… Mot clé : car(char)

d) Type chaînes de caractères : concerne des chaines de caractères tels que des mots ou des phrases : ”informatique”, ”la section B”,… Mot clé : chaîne(string)

e) Type booléen : ce type ne peut prendre que deux états : vrai ou faux Mot clé : booléen(boolean)

Les types d'une variable en c

- Voici les principaux types de variables existant en langage C :

Nom du type	Taille (octet)	minimum	Maximum
int	4	–2 147 483 648	2 147 483 647
long	8	–9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned int	4	0	4 294 967 295
float	4	-3.40282×10^{38}	3.40282×10^{38}
double	8	-1.79769×10^{308}	1.79769×10^{308}
char	1	-127	127

- Pour un nombre entier, on utilisera le plus souvent int pour un nombre flottant (ou réel ou décimal), on utilisera généralement double.

Instructions de base

Les principales instructions sur les variables en algorithmique sont :

- La déclaration d'une variable
- L'affectation
- L'initialisation d'une variable
- La lecture d'une variable
- L'affichage d'une variable

Déclaration d'une variable

Avant d'utiliser une variable dans un algorithme ou dans un programme, il faut la créer autrement dit la déclarer. Il suffit juste :

1. d'indiquer le type de la variable qu'on veut créer ;
2. d'insérer un espace ;
3. d'indiquer le nom que vous voulez donner à la variable ;
4. et enfin, de ne pas oublier le point-virgule.

Déclaration d'une variable

Exemple en pseudo-code :

```
Exemple de déclaration
Var
    MoyenneEtudiant: réel;
    IdentifiantEtudiant: entier;
    Début


    Fin
```

Exemple en langage C :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    double MoyenneEtudiant;
    int IdentifiantEtudiant;
    return 0;
}
```

Affectation

- Affecter une valeur à une variable c'est remplir cette variable avec une valeur.
- Pour affecter une valeur à une variable, indiquez simplement le nom de la variable, puis le signe égal (=) en langage C ou la flèche en pseudo code (←), et enfin la valeur que vous voulez y mettre.

Exemple en pseudo code:
NoteAlgo←14,5;

Exemple en langage C:
NoteAlgo=14,5;

Initialisation d'une variable

- Initialiser une variable c'est la déclarer et lui donner une valeur initiale.
- L'avantage, c'est que vous êtes sûrs que cette variable contient une valeur correcte, et pas du n'importe quoi.

Exemple en pseudo-code :

```
Exemple de déclaration
Var
    MoyenneEtudiant: réel;
    IdentifiantEtudiant: entier;
    Début
    MoyenneEtudiant ← 12,5;
    IdentifiantEtudiant ← 145;
    Début
    Fin
```

Exemple en langage C :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   double MoyenneEtudiant=12,5;
   int IdentifiantEtudiant=145;
   return 0;
}
```

L'affichage en pseudo code

- L'Instruction responsable de l'affichage en pseudo code est **Afficher ou Écrire.**
- Par exemple pour afficher le message bonjour à l'écran de l'utilisateur, il suffit: Afficher (" bonjour ")
- Pour afficher la valeur d'une variable par exemple:

A ← 3

Afficher (A)

- Pour afficher la valeur d'une variable avec un message:

Afficher (" la valeur de la variable " , A)

L'affichage dans le langage C

- L'Instruction responsable de l'affichage dans le langage C est **printf**
- Par exemple pour afficher le message bonjour à l'écran de l'utilisateur, il suffit: printf(" bonjour ")
- Pour afficher la valeur d'une variable il faut ajouter par exemple:

```
int A=3;
printf(" %d",A);
```

La lettre après le % permet d'indiquer ce qu'on doit afficher. 'd' signifie que l'on souhaite afficher un int.

L'affichage dans le langage C

Type	format
int	%d
long	%ld
float	%f
double	%lf
char	%c

- Pour afficher la valeur d'une variable avec un message:

```
int A=3;
printf (" la valeur de la variable est de %d " , A);
```

Afficher plusieurs variables avec un seul printf

- Il est possible d'afficher la valeur de plusieurs variables dans un seul printf.
- Il vous suffit pour cela 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.

Exemple :

```
 int main() {
int nombreEtudiants = 240;
double moyenneBac = 14.5 ;
printf("Il y a %d etudiants inscrits avec une moyenne de %lf en Bac", nombreEtudiants, moyenneBac);
 return 0;}
```

Afficher plusieurs variables en pseudo code

Exemple en pseudo-code :

Exemple de déclaration

Var

MoyenneEtudiant: réel;
IdentifiantEtudiant: entier;
Début
MoyenneEtudiant ← 12,5;
IdentifiantEtudiant ← 145;
Début
Écrire ("l'etudiant numero" , IdentifiantEtudiant, "a eu une moyenne de" , MoyenneEtudiant)
Fin

Lecture (saisie) d'une variable

- L'instruction pour la saisie des valeurs des variables en pseudo code est **lire**.
- Pour saisir la valeur de la variable a, il suffi: lire(a)
- L'instruction pour la saisie des valeurs des variables dans le langage c est **scanf**.
- Cette fonction ressemble beaucoup à printf. Vous devez mettre un format pour indiquer ce que l'utilisateur doit entrer (un int, un float, ...). Puis vous devez indiquer le nom de la variable qui va recevoir le nombre.

Lecture (saisie) d'une variable

Exemple :

```
 int main() {

int age = 0;

scanf("%d", &age);

return 0; } .
```

- On doit mettre le %d entre guillemets. Par ailleurs, il faut mettre le symbole & devant le nom de la variable qui va recevoir la valeur

Opérations arithmétiques

Opérateur	Opération	Type des opérandes	Type du résultat
+	L'addition	entier/réel	entier/réel
-	La soustraction	entier/réel	entier/réel
*	La multiplication	entier/réel	entier/réel
/	La division	entier/réel	entier/réel
%	Le reste de la division entière	entier	entier
Fmod()	Le reste de la division réelle	réel	réel

Opérateurs d'affectation

Opérateur	*Signification*	*Exemple*
=	Affectation ordinaire	X=Y
+=	ajouter à	X+=Y X=X+Y
-=	diminuer de	X-=Y X=X-Y
*=	multiplier par	X*=Y X=X*Y
/=	diviser par	X/=Y X=X/Y
%=	modulo	X%=Y X=X%Y
--	Décrémentation de 1	X-- X=X-1
++	Incrémentation de 1	X++ X=X+1
X = i++	passe d'abord la valeur de i à X et incrémente après	X= i i =i+1
X = i--	passe d'abord la valeur de i à X et décrémente après	X= i i =i-1
X = ++i	incrémente d'abord et passe la valeur incrémentée à X	i =i+1 X=i
X = --i	décrémente d'abord et passe la valeur décrémentée à X	i =i-1 X=i

Exemple pour conclure

Ecrire un petit algorithme qui demande l'âge de l'utilisateur et qui affiche cet âge et ensuite de le traduire en programme C.

Structuration d'un algorithme

Algorithme nom_de_l'algorithme

Var nom_variable : type_variable;

Const nom_constante=valeur;

Début

Action 1 ;

Action 2 ;

……. <Partie Instructions>

Action n ,

Fin

L'en-tête

Les déclarations

Le corps

Exemple pour conclure

Ecrire un petit algorithme qui demande l'âge de l'utilisateur et qui affiche cet âge et ensuite de le traduire en programme C.

Algorithme saisie et affichage de l'âge

Var age:entier;

début

Afficher(" Quel age avez-vous ? ");

Lire(age) ;

Afficher (" Ah ! Vous avez donc " , age, " ans ");

Fin

Structuration d'un programme en C

```
# include <stdio .h>

# include <stdlib .h>

int main ()

{

// instruction

printf (" Salam ");

return 0;

}
```

Bibliothèques

Début du programme

Commentaire

Instruction qui permet d'afficher Salam

Fin du programme

Corps du programme

Traduction de l'exemple en langage C

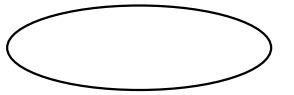
```
# include <stdio .h>
# include <stdlib .h>
int main()
 {
printf("Quel age avez-vous ? ");
scanf("%d", &age); // On demande d'entrer l'age avec scanf
printf("Ah ! Vous avez donc %d ans !\n", age);
 return 0;
}
```

Exercice

Trouver l'organigramme qui permet de calculer la somme de deux entiers sachant que ces deux entiers sont saisis par l'utilisateur, ensuite le traduire en programme c.

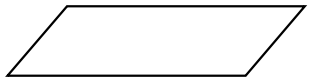
L'organigramme

L'organigramme est une représentation graphique de l'enchaînement d'une suite d'actions. Un organigramme utilise des symboles graphiques.

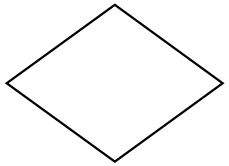


Le début et la fin d'un traitement

Les opérations de traitement



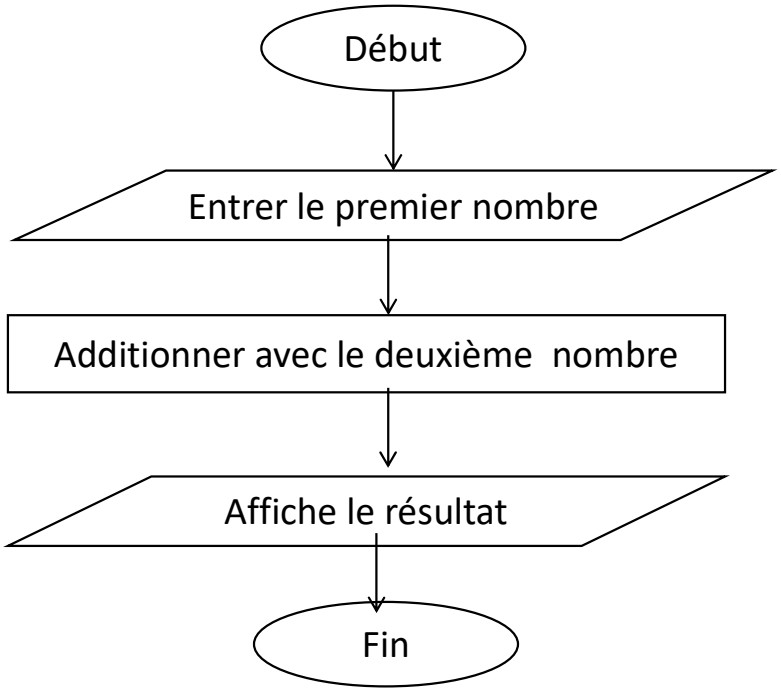
Une entrée ou une sortie



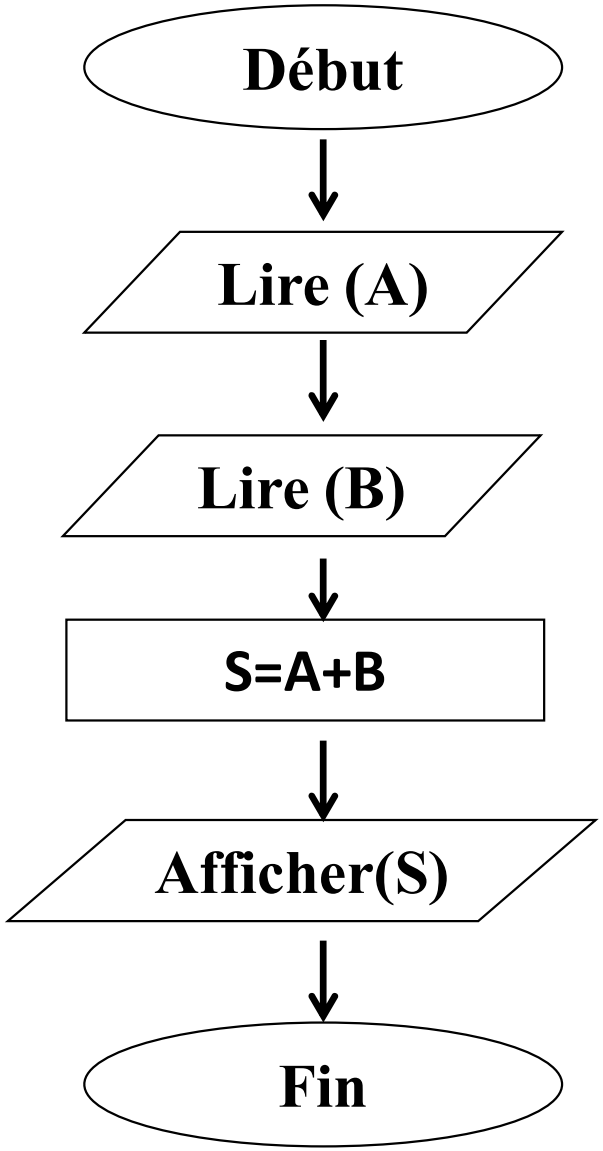
Test (condition)

Sens des actions

Exemple de l'addition avec la calculatrice



L'organigramme



Structuration d'un programme en C

```
# include <stdio .h>

# include <stdlib .h>

int main ()

{

// instruction

printf (" Salam ");

return 0;

}
```

Bibliothèques

Début du programme

Commentaire

Instruction qui permet d'afficher Salam

Fin du programme

Corps du programme

Somme de deux entiers

```
# include <stdio .h>

# include <stdlib .h>
int main() {
1.  int resultat = 0, nombre1 = 0, nombre2 = 0; //initialisation des variables
2.   // On demande les nombres 1 et 2 a l'utilisateur
3.  printf("Entrez le nombre 1 : ");
4.   scanf("%d", &nombre1);
5.  printf("Entrez le nombre 2 : ");
6.  scanf("%d", &nombre2);
7.  resultat = nombre1 + nombre2; // On fait le calcul
8.   // Et on affiche l'addition a l'ecran :
9.  printf ("%d + %d = %d\n", nombre1, nombre2, resultat);
10. return 0;}
```

Trace d'exécution du programme de la somme des deux entiers

Dans le tableau de la trace (historique) d'exécution, nous allons voire le changement des valeurs des variable en mémoire du début de l'exécution jusqu''à la fin.

On suggère que les valeurs entrées par l'utilisateur **pour le nombre1** est **6** et pour **le nombre2** est **4.**

	nombre1	**nombre2**	**résultat**
Instruction 1	0	0	0
Instruction 4	6	0	0
Instruction 6	6	4	0
Instruction 7	6	4	10

Exercice sur la trace d'exécution

Remplir les commentaires et trouver l'historique d'exécution et l'affichage du programme suivant:

```
#include <stdio.h>
int main() {
1.      int a = 5, b = 10, c = 15, d = 20, i=0; // …….
2.      a += 3; //……
3.      b -= 4; // ……
4.      c *= 2; // ……
5.      d = c++; // ……
6.      a += b; // ……
7.      c -= d; // ……
8.       i++; // ……
9.       b = --i; // ……
10.     d %= 3; // ……
11.     printf("Valeurs finales : a = %d, b = %d, c = %d, d = %d\n,i= %d\n", a, b, c, d,i);
12. return 0;}
```

Exercice sur la trace d'exécution

Remplissage des commentaires :

```
#include <stdio.h>
int main() {
1.      int a = 5, b = 10, c = 15, d = 20,i=0; // initialisation de cinq variables
2.      a += 3; //a=a+3
3.      b -= 4; // b=b-4
4.      c *= 2; // c=c*2
5.      d = c++; //  d=c, c=c+1
6.      a += b; // a=a+b
7.      c -= d; //c=c-d
8.      i++; // incrémentation de i, i=i+1
9.      b = --i; // i=i-1 b=i
10.     d %= 3; // d=d%3
11.     printf("Valeurs finales : a = %d, b = %d, c = %d, d = %d\n,i= %d\n", a, b, c, d, i);
12.     return 0;
}
```

Exercice sur la trace d'exécution

Le tableau de l'historique d'exécution :

	a	b	c	d	i
Instruction 1	**5**	**10**	**15**	**20**	**0**
Instruction 2	**8**	**10**	**15**	**20**	**0**
Instruction 3	**8**	**6**	**15**	**20**	**0**
Instruction 4	**8**	**6**	**30**	**20**	**0**
Instruction 5	**8**	**6**	**31**	**30**	**0**
Instruction 6	**14**	**6**	**31**	**30**	**0**
Instruction 7	**14**	**6**	**1**	**30**	**0**
Instruction 8	**14**	**6**	**1**	**30**	**1**
Instruction 9	**14**	**0**	**1**	**30**	**0**
Instruction 10	**14**	**0**	**1**	**0**	**0**

Exercice sur la trace d'exécution

L'affichage finale: