

Algorithmique



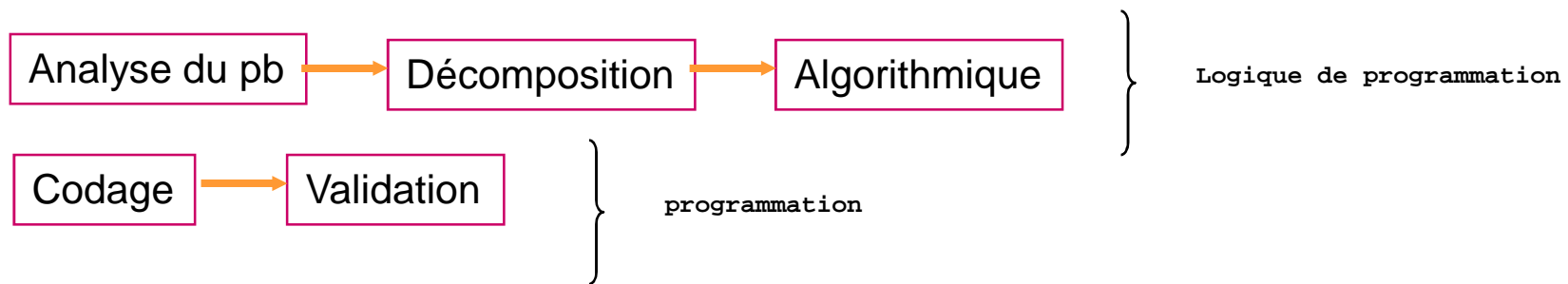
Algorithmique et Programmation en C

Chapitre 2

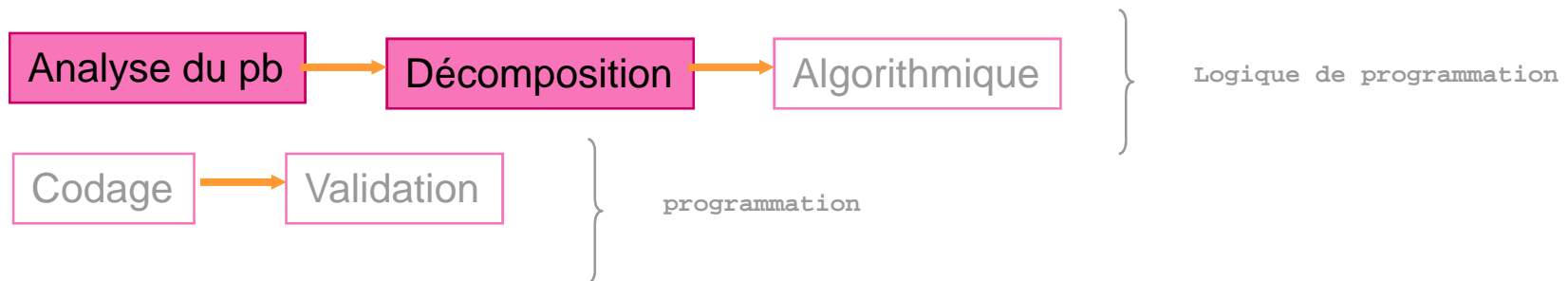
Karim Bouzoubaa

Objectif de l'ingénierie

- Résoudre des problèmes de divers types
- Objectif du cours
 - Trouver et développer des solutions techniques (informatiques) d'un problème donné

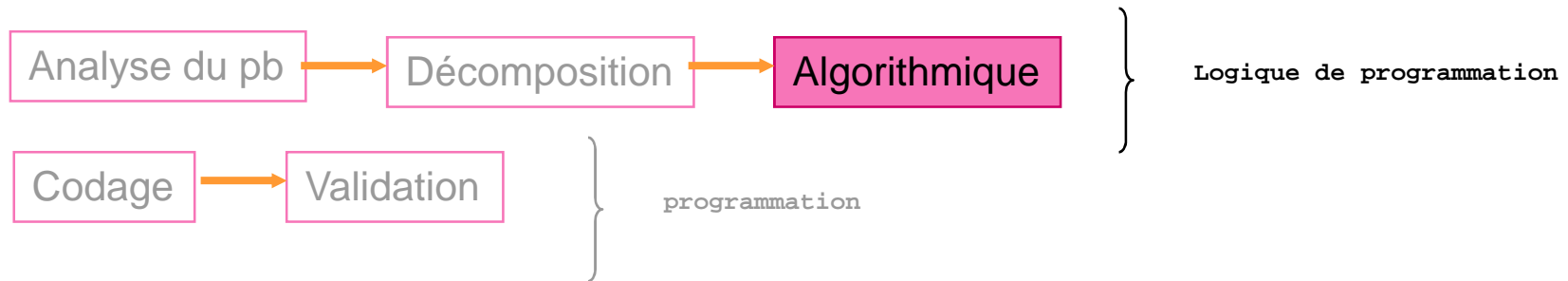


Analyse & Décomposition



- Analyse
 - Étape de recherche de la méthode de résolution
 - Étape la plus importante
 - Lorsque bien faite, facilite les étapes qui suivent
- Décomposition
 - C'est un moyen de vaincre la complexité
 - procéder par une décomposition successive
 - Décomposer à partir du général pour aller aux détails

Algorithmique & Logique



- *Objectif de l'algorithmique*
 - Code standard (traduire vers n'importe quel langage)
- *Logique de programmation*
 - Solution technique d'un problème donné
 - Enchaînement logique de différentes étapes à suivre pour résoudre un pb (étape_{n+1} après étape_n)
 - une liste d'instructions permettant de construire un programme

Définition



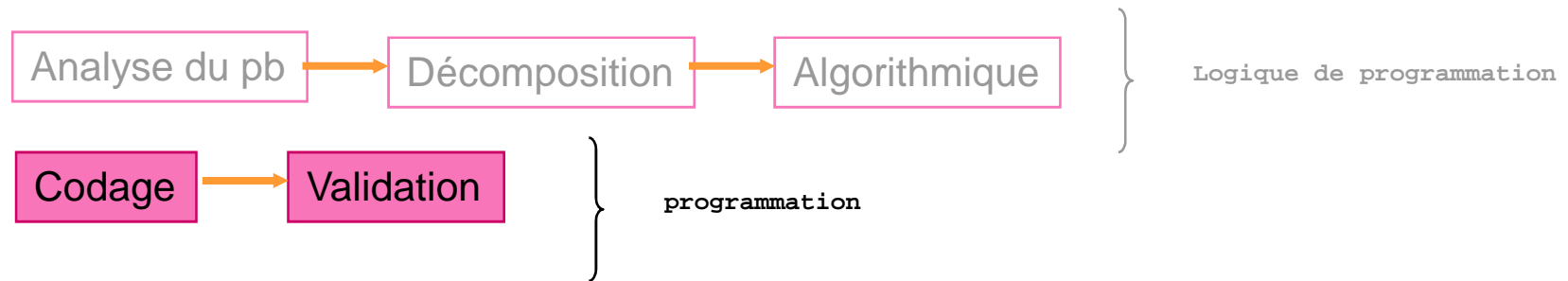
- *Algorithme*
 - Suite finie d'instructions
 - Ordre déterminé
 - Appliqué à un nombre fini de données
 - Indépendant de l'ordinateur et des langages de programmation
- *Exemple: Jardinage*
 - Actions de base:
 - Creuser un trou
 - Reboucher un trou
 - Placer un arbre
 - Arroser

Exemple



- *Algorithme de plantation d'un arbre*
 - 1- Creuser un trou
 - 2- Placer un arbre dans le trou
 - 3- Reboucher le trou
- *Algorithme de plantation et d'arrosage de plusieurs arbres:*
 - 1- Creuser un trou
 - 2- Placer un arbre dans le trou.
 - 3- Reboucher le trou
 - 4- S'il existe encore des arbres
 - Exécuter les actions 1, 2, 3 et 4
 - Sinon Exécuter les actions suivantes
 - 5- Arroser les arbres
- *Remarques:*
 - Planter tous les arbres et les arroser à la fin
 - ou*
 - Planter et arroser arbre par arbre
 - A un problème donné pourraient correspondre **plusieurs** algorithmes
 - Les différences résident dans la complexité (espace, temps)

Codage & Validation



- Codage
 - Transcrire la résolution du pb dans un langage de programmation
- Validation
 - Tester le programme à l'aide d'un jeu de tests

Synthèse – Étapes de résolution d'un pb

1. Analyser

- Décomposer
- Préciser (Données, Résultats, Liens entre eux)

2. Algorithmique

- Plusieurs algo sont généralement possibles
- Les actions doivent être précises
- Un algo doit pouvoir se terminer et donner le résultat attendu
- L'algo doit être indépendant de tout langage de programmation (écrit en langage naturel)

3. Traduction

- Respecter les règles d'écriture du langage choisi
- Respecter le découpage détaillé dans l'algo
- Commenter le prog

4. Exécuter et tester le programme

- Exécuter le programme avec une série d'exemples pour tester s'il est juste
- Types d'erreurs:
 - de syntaxe (d'écriture)
 - de logique (revoir l'algo)
 - d'exécution

5. Documenter le prog

- Spécifier: le rôle du prog, le type de données et des résultats, l'algo utilisé

Exemple



- **Pb**: Trouver le max de trois variables x , y et z
- **Analyse**: plusieurs stratégies possibles
 - Déterminer le max de deux variables et le comparer à la troisième
 - Comparer chaque variable avec les deux autres en même temps
 - etc.

 - Données: x , y et z
 - Résultats: $\max = \text{maximum des données}$

Exemple – suite 1

- **Algorithme**

- Algorithme "maximum de 3 variables"
- Entrées : x, y, z type entier
- Sorties : max type entier
- Temporaires: max_temp
- Traitement:
 - Début
 - Lire x, y, z

 - si ($x > y$) alors max_temp = x
 - sinon max_temp = y

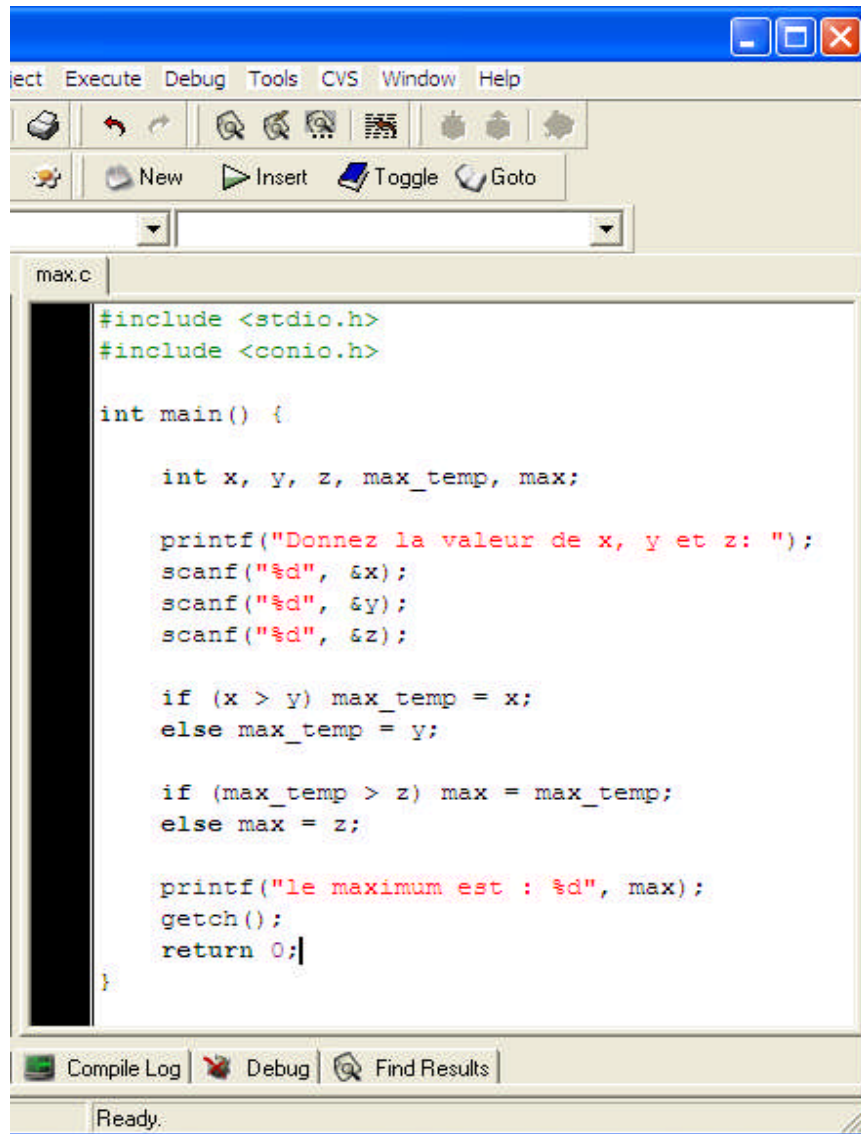
 - si (max_temp > z) alors max = max_temp
 - sinon max = z

 - Afficher(" le maximum est : ", max)

 - Fin

Exemple – suite 2

- Traduction: en C & Java



The screenshot shows a text editor window titled 'max.c'. The code is a C program that finds the maximum of three integers x, y, and z. It uses printf for output and scanf for input. The logic uses nested if-else statements to compare the values.

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

int main() {

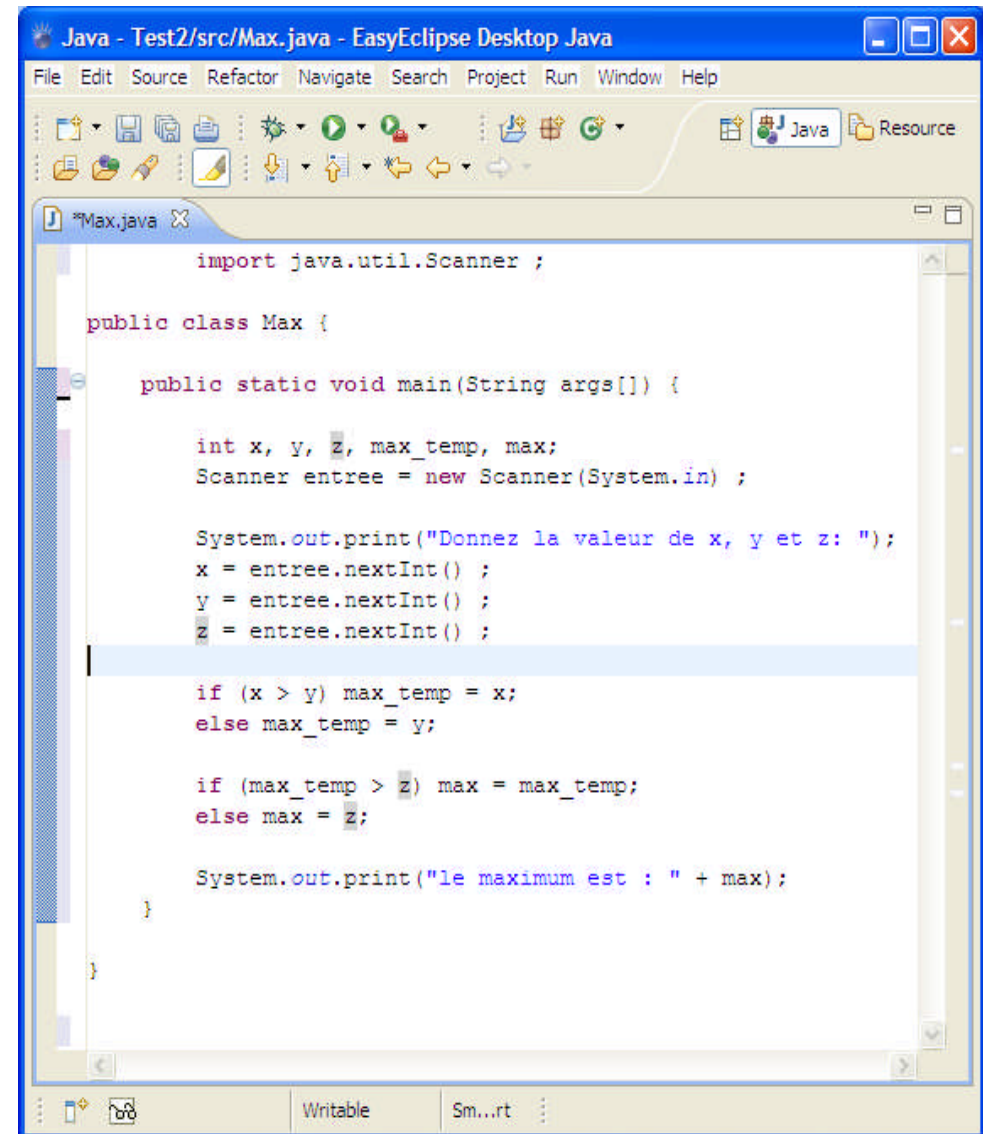
    int x, y, z, max_temp, max;

    printf("Donnez la valeur de x, y et z: ");
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);

    if (x > y) max_temp = x;
    else max_temp = y;

    if (max_temp > z) max = max_temp;
    else max = z;

    printf("le maximum est : %d", max);
    getch();
    return 0;
}
```



The screenshot shows the Eclipse IDE with a Java file named 'Max.java'. The code is a Java program that finds the maximum of three integers x, y, and z. It uses Scanner for input and System.out.print for output. The logic uses nested if-else statements to compare the values.

```
import java.util.Scanner ;

public class Max {

    public static void main(String args[] ) {

        int x, y, z, max_temp, max;
        Scanner entree = new Scanner(System.in) ;

        System.out.print("Donnez la valeur de x, y et z: ");
        x = entree.nextInt() ;
        y = entree.nextInt() ;
        z = entree.nextInt() ;

        if (x > y) max_temp = x;
        else max_temp = y;

        if (max_temp > z) max = max_temp;
        else max = z;

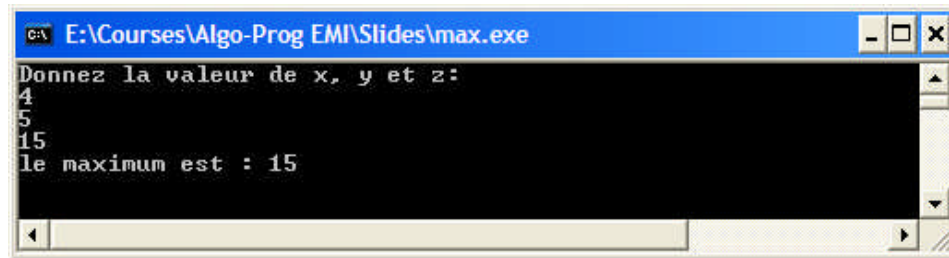
        System.out.print("le maximum est : " + max);

    }

}
```

Exemple – suite 3

- Exécution et Tests



```
C:\E:\Courses\Algo-Prog EM\Slides\max.exe
Donnez la valeur de x, y et z:
4
5
15
le maximum est : 15
```

```
int main() {

    int x, y, z, max_temp, max;

    printf("Donnez la valeur de x, y et z: ");
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);

    if (x > y) max_temp = x;
    else max_temp = y;

    if (max_temp > z) max = max_temp;
    else max = z;

    printf("le maximum est : %d", max);
```

mpile Log |  Debug |  Find Results |  Close

Message

[Linker error] undefined reference to `printf'

Exemple – suite 4

- Documentation

```
// Ce programme consiste à déterminer le maximum de trois
// variables entières. L'algorithme consiste à déterminer
// d'abord le maximum temporaire des deux premières variables
// et à comparer ensuite ce maximum temporaire avec la troisième
// variable

#include <stdio.h>
#include <conio.h>

int main() {

    // Données, Résultats et variables temporaires
    int x, y, z, max_temp, max;

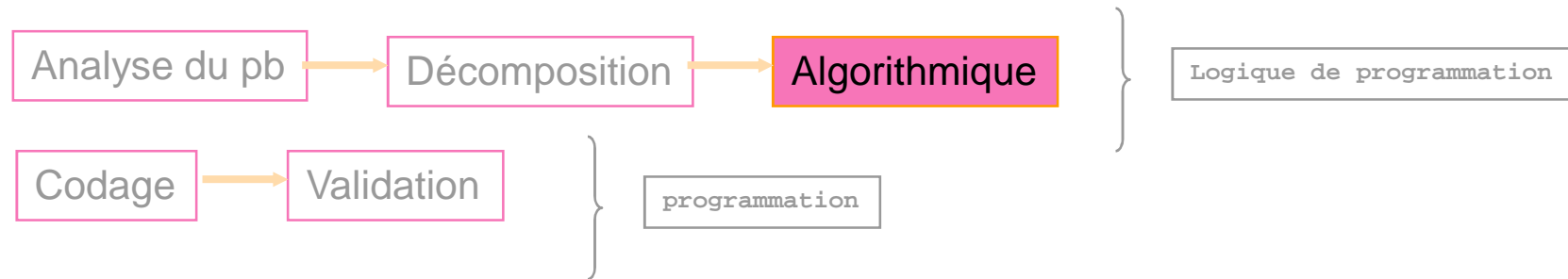
    // Lecture des trois variables en entrée
    printf("Donnez la valeur de x, y et z: ");
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &z);

    // Déterminer la variable temporaire
    if (x > y) max_temp = x;
    else max_temp = y;

    // Comparaison avec la troisième variable
    if (max_temp > z) max = max_temp;
    else max = z;          int max_temp

    // Affichage des résultats
    printf("le maximum est : %d", max);
    getch();
    return 0;
}
```

Objectif du chapitre



- La suite du présent chapitre consiste à étudier en détails la partie **ALGORITHMES** dans le cycle de développement de la résolution d'un pb

Composantes d'un algorithme

- Algorithme "maximum de 3 variables"
- Entrées : x, y, z type entier
- Sorties : max type entier
- Temporaires: max_temp
- Traitement:
 - Début
 - Lire x, y, z
 - si (x > y) alors max_temp = x
 - sinon max_temp = y
 - si (max_temp > z) alors max = max_temp
 - sinon max = z
 - Afficher(" le maximum est : ", max)
 - Fin

- **Entête**
 - Le nom - utilité
 - Données
 - Entrées
 - Sorties
 - Auxiliaires
- **Corps**
 - Mot clef Début
 - Instructions
 - Mot clef Fin

Exemple d'Algorithme 1

Algorithme "calcul de la somme $S = 1 + 2 + \dots + n$ "

Entrées : n type entier

Sorties : s type entier

Traitement:

 Début

 Lire n

$s = n * (n + 1) / 2$

 Afficher(" la somme est : ", s)

 Fin

Exemple d'Algorithme 2

Algorithme "somme, produit et division de deux variables"

Entrées : x, y type entier

Sorties : s, p, d type entier

Traitement:

 Début

 Lire x, y

 s = x + y

 p = x * y

 d = x / y

 Afficher(" la somme est : ", s)

 Afficher(" le produit est : ", p)

 Afficher(" la division est : ", d)

 Fin

- Discussion
 - Erreur d'exécution
 - Nomination des variables
 - Lisibilité
 - Indentation

Nomination & Lisibilité

Algorithme "somme, produit et division de deux variables"

Entrées : x, y type entier

Sorties : s, p, d type entier

Traitement:

 Début

 Lire x, y

 p = x + y

 matecha = x * y

 s = x / y

 Afficher(" la somme est : ", p)

 Afficher(" le produit est : ", matecha)

 Afficher(" la division est : ", s)

 Fin

Indentation



```
Algorithme "somme, produit et
division de deux variables" Entrées : x, y type entier
Sorties : s, p, d type entier Traitement:
    Début Lire x, y
s = x + y
    p = x * y      d = x / y
    Afficher(" la somme est : ", s)
    Afficher(" le produit est : ", p)
Afficher(" la division est : ", d) Fin
```

Bonnes pratiques

Algorithme "somme, produit et division de deux variables"

Entrées : x, y type entier

Sorties : somme, produit, division type entier

Traitement:

 Début

 Lire x, y

 somme = x + y

 produit = x * y

 division = x / y

 Afficher(" la somme est : ", somme)

 Afficher(" le produit est : ", produit)

 Afficher(" la division est : ", division)

 Fin

Prochains chapitres

- Étude de la partie **Données**
 - **Structure de données** pouvant être utilisées dans un algo et dans un programme
- Étude de la partie **Instructions**
 - **Structures de contrôle** pouvant être utilisées dans un algo et dans un programme

- Entête
 - Le nom
 - **Données**
 - Entrées
 - Sorties
 - Auxiliaires
- Corps
 - Début
 - **Instructions**
 - Fin

Exercices



1. Somme, Produit et division de trois variables
2. Calcul de la note finale d'un étudiant. L'étudiant possède 3 notes (TP, sem1 et sem2), note finale est calculée avec pondération resp de 20%, 30% et 50%
3. Calcul de la distance entre deux points $P1(x1, y1)$ et $P2(x2, y2)$
4. Lecture d'un nombre de trois chiffres et extraction du chiffre des unités, du chiffre des dizaines et du chiffre des centaines