

COBOL - Exercices et Corrections Complètes

Tous les exercices du cours COBOL avec corrections détaillées, explications ligne par ligne et variantes

Table des Matières

- [Partie 1 : Fondamentaux](#)
 - [Exercice 1 : Questions de Compréhension](#)
 - [Exercice 2 : Architecture des LLMs](#)
 - [Exercice 3 : Vertex AI Pratique](#)
 - [Partie 2 : Programmation de Base](#)
 - [Exercice 4 : Opérations Arithmétiques](#)
 - [Exercice 5 : Manipulation de Chaînes](#)
 - [Exercice 6 : Structures Conditionnelles](#)
 - [Exercice 7 : Boucles](#)
 - [Exercice 8 : Gestion de Fichiers](#)
 - [Partie 3 : Fichiers Avancés](#)
 - [Exercice 9 : Fichiers Indexés](#)
 - [Exercice 10 : Tables et Tableaux](#)
 - [Projets Complètes](#)
 - [Projet 1 : Système Bancaire](#)
 - [Projet 2 : Gestion de Stock](#)
 - [Projet 3 : Paie des Employés](#)
-

Partie 1 : Fondamentaux

Exercice 1 : Questions de Compréhension

Énoncé

Réponds aux questions suivantes pour vérifier ta compréhension des concepts de base :

1. Quelle est la différence principale entre IA discriminative et générative ?
2. Qu'est-ce qu'un token dans le contexte des LLMs ?
3. Comment la température affecte-t-elle la génération de texte ?
4. Donnez 3 exemples d'applications pratiques de l'IA générative
5. Qu'est-ce qu'une "hallucination" dans un LLM ?
6. Pourquoi dit-on que les LLMs sont entraînés de manière "auto-supervisée" ?
7. Quelle est l'utilité d'une context window large ?

✅ Corrections

Question 1 : Différence IA discriminative vs générative

Réponse :

- **IA Discriminative** : Classifie ou prédit une catégorie à partir de données existantes (ex: "Est-ce un chat ou un chien ?")
- **IA Générative** : Crée du nouveau contenu original (ex: "Génère une image d'un chat")

Exemple concret :

Discriminative : Photo → "C'est un chat" ✅

Générative : "Chat roux jouant" → 🎨 Génère l'image

Question 2 : Token

Réponse : Un token est une unité de base du texte pour les LLMs. Approximativement :

- 1 token \approx 4 caractères en anglais
- 1 token \approx 0.75 mots
- "Hello world!" \approx 3 tokens

Importance : Les LLMs ont des limites en tokens (context window), donc comprendre les tokens aide à optimiser les prompts.

Question 3 : Température

Réponse : La température contrôle le niveau de créativité/aléatoire :

- **0.0-0.3** : Déterministe, répétitif, prévisible (pour classification)
- **0.7** : Équilibré (défaut)
- **1.0+** : Créatif, varié, imprévisible (pour brainstorming)

Question 4 : Applications pratiques

Réponse :

1. **Génération de contenu** : Articles, posts réseaux sociaux
2. **Assistance au code** : GitHub Copilot, génération de code
3. **Service client** : Chatbots intelligents
4. **Création artistique** : Génération d'images (Midjourney, DALL-E)
5. **Traduction** : Traduction contextuelle avancée

Question 5 : Hallucination

Réponse : Une hallucination = quand le LLM invente des informations fausses mais plausibles.

Exemple :

Prompt : "Cite l'étude de l'université de Stanford sur..."

LLM : "Selon l'étude de Stanford (2023)..."

✗ L'étude n'existe pas !

Question 6 : Auto-supervisé

Réponse : Les LLMs apprennent en prédisant le mot suivant dans un texte, sans labels humains :

Texte : "Le chat dort sur le..."

Tâche : Prédire "canapé"

Répété des milliards de fois → apprentissage du langage

Question 7 : Context window large

Réponse : Avantages :

- Comprendre des documents longs
- Conversations plus longues
- Meilleure cohérence
- Analyser des fichiers entiers

Exemple :

- GPT-3.5 : 4K tokens (~3000 mots)
- Claude 2 : 100K tokens (~75000 mots) = peut lire un livre entier !

Exercice 2 : Premier Programme Simple

Énoncé

Crée un programme COBOL qui :

1. Demande ton nom
2. Demande ton âge
3. Affiche un message personnalisé : "Bonjour [NOM], tu as [ÂGE] ans!"
4. Si l'âge ≥ 18 , affiche "Tu es majeur"
5. Sinon, affiche "Tu es mineur"

Solution Complète

* Programme : EXERCICE2-INFO-PERSONNELLES.cob

* Description : Demande nom et âge, affiche message personnalisé

* Auteur : Correction Exercice 2

* Date : 02/11/2024

IDENTIFICATION DIVISION.

PROGRAM-ID. EX2-INFO-PERSO.

AUTHOR. VOTRE-NOM.

DATE-WRITTEN. 02/11/2024.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

*--- Variables pour stocker les informations ---

01 WS-NOM PIC X(30).

01 WS-AGE PIC 999.

*--- Variables pour l'affichage ---

01 WS-LIGNE-SEPARATION PIC X(50) VALUE ALL "=".

*--- Conditions avec niveau 88 ---

01 WS-STATUT-MAJORITE PIC X.

88 MAJEUR VALUE "M".

88 MINEUR VALUE "m".

PROCEDURE DIVISION.

*-----

* Programme principal

*-----

000-MAIN-PROCEDURE.

PERFORM 100-AFFICHER-ENTETE.

PERFORM 200-SAISIR-INFORMATIONS.

PERFORM 300-AFFICHER-RESULTATS.

PERFORM 400-VERIFIER-MAJORITE.

PERFORM 900-TERMINER.

STOP RUN.

*-----

* Afficher l'en-tête du programme

*-----

100-AFFICHER-ENTETE.

DISPLAY " ".

DISPLAY WS-LIGNE-SEPARATION.

DISPLAY " INFORMATIONS PERSONNELLES ".

DISPLAY WS-LIGNE-SEPARATION.

DISPLAY " ".

*-----

* Saisir le nom et l'âge

*-----

200-SAISIR-INFORMATIONS.

*--- Demander le nom ---

DISPLAY "Quel est ton nom ? " WITH NO ADVANCING.

ACCEPT WS-NOM.

*--- Demander l'âge ---

DISPLAY "Quel est ton age ? " WITH NO ADVANCING.

ACCEPT WS-AGE.

DISPLAY " ".

*-----

* Afficher le message personnalisé

*-----

300-AFFICHER-RESULTATS.

DISPLAY WS-LIGNE-SEPARATION.

DISPLAY " ".

DISPLAY "Bonjour " WS-NOM ", tu as " WS-AGE " ans!".

DISPLAY " ".

*-----

* Vérifier si majeur ou mineur

*-----

400-VERIFIER-MAJORITE.

IF WS-AGE >= 18

SET MAJEUR TO TRUE

DISPLAY "Tu es MAJEUR."

ELSE

SET MINEUR TO TRUE

DISPLAY "Tu es MINEUR."

END-IF.

DISPLAY " ".

*-----

* Terminer proprement

*-----

900-TERMINER.

DISPLAY WS-LIGNE-SEPARATION.

DISPLAY "Programme termine. Au revoir!".

DISPLAY WS-LIGNE-SEPARATION.

DISPLAY " ".

Explications Ligne par Ligne

IDENTIFICATION DIVISION :

PROGRAM-ID. EX2-INFO-PERSO.

- Nom du programme (30 caractères max)
- Utilisé pour identifier le programme

DATA DIVISION - Variables :

01 WS-NOM PIC X(30).

- WS- : Préfixe Working-Storage
- PIC X(30) : Alphanumérique, 30 caractères maximum
- Stockera le nom saisi

01 WS-AGE PIC 999.

- PIC 999 : Numérique, 3 chiffres (0-999)
- Stockera l'âge

01 WS-LIGNE-SEPARATION PIC X(50) VALUE ALL "=".

- VALUE ALL "=" : Remplit avec 50 caractères "="
- Utilisé pour les séparateurs visuels

Niveau 88 (Conditions) :

01 WS-STATUT-MAJORITE PIC X.

88 MAJEUR VALUE "M".

88 MINEUR VALUE "m".

- Alternative élégante aux IF multiples
- SET MAJEUR TO TRUE = MOVE "M" TO WS-STATUT-MAJORITE
- Plus lisible dans le code

PROCEDURE DIVISION - Structure :

000-MAIN-PROCEDURE.

PERFORM 100-AFFICHER-ENTETE.

PERFORM 200-SAISIR-INFORMATIONS.

...

- Structure modulaire avec PERFORM
- Numérotation : 000 (main), 100, 200, etc. (sous-procédures)
- Facile à suivre et maintenir

Saisie avec ACCEPT :

DISPLAY "Quel est ton nom ? " WITH NO ADVANCING.

ACCEPT WS-NOM.

- WITH NO ADVANCING : Pas de retour à la ligne (saisie sur même ligne)

- ACCEPT : Lit depuis le clavier et stocke dans WS-NOM

Condition IF :

IF WS-AGE >= 18

SET MAJEUR TO TRUE

DISPLAY "Tu es MAJEUR."

ELSE

SET MINEUR TO TRUE

DISPLAY "Tu es MINEUR."

END-IF.

- >= : Supérieur ou égal
- SET ... TO TRUE : Utilise le niveau 88
- END-IF : Ferme le bloc IF

Exemple d'Exécution

=====

INFORMATIONS PERSONNELLES

=====

Quel est ton nom ? Jean Dupont

Quel est ton age ? 25

=====

Bonjour Jean Dupont, tu as 25 ans!

Tu es MAJEUR.

=====

Programme termine. Au revoir!

=====

Variantes Possibles

Variante 1 : Sans niveau 88


```
IF WS-AGE >= 18  
    DISPLAY "Tu es MAJEUR."  
ELSE  
    DISPLAY "Tu es MINEUR."  
END-IF.
```


Variante 2 : Avec EVALUATE (plus de catégories)

```
EVALUATE TRUE  
    WHEN WS-AGE < 13  
        DISPLAY "Tu es un enfant."  
    WHEN WS-AGE >= 13 AND WS-AGE < 18  
        DISPLAY "Tu es un adolescent."  
    WHEN WS-AGE >= 18 AND WS-AGE < 60  
        DISPLAY "Tu es un adulte."  
    WHEN WS-AGE >= 60  
        DISPLAY "Tu es senior."  
END-EVALUATE.
```

Pièges à Éviter

1. Oublier END-IF

 IF WS-AGE >= 18
 DISPLAY "Majeur"
ELSE
 DISPLAY "Mineur". *> Pas de END-IF !

 IF WS-AGE >= 18
 DISPLAY "Majeur"

ELSE

DISPLAY "Mineur"

END-IF. *> Correct

2. Mauvais type de PICTURE

✗ 01 WS-AGE PIC X(3). *> Alphanum pour un nombre !

✓ 01 WS-AGE PIC 999. *> Numérique correct

3. Oublier WITH NO ADVANCING

✗ DISPLAY "Nom ?".

ACCEPT WS-NOM.

*> Résultat :

*> Nom ?

*> [cursor sur nouvelle ligne]

✓ DISPLAY "Nom ?" WITH NO ADVANCING.

ACCEPT WS-NOM.

*> Résultat :

*> Nom ? [cursor après ?]

🎓 Points Clés à Retenir

1. ✓ **Structure PERFORM** : Organise le code en modules logiques
2. ✓ **ACCEPT** : Lit depuis l'entrée standard (clavier)
3. ✓ **DISPLAY** : Écrit vers la sortie standard (écran)
4. ✓ **WITH NO ADVANCING** : Garde le curseur sur la même ligne
5. ✓ **Niveau 88** : Rend le code plus lisible
6. ✓ **IF...ELSE...END-IF** : Structure conditionnelle complète

Exercice 3 : Calculatrice Basique

 **Énoncé**

Crée une calculatrice COBOL qui :

1. Demande deux nombres
2. Affiche un menu avec 4 opérations (+, -, ×, ÷)
3. Demande le choix de l'utilisateur
4. Effectue le calcul
5. Affiche le résultat
6. Gère la division par zéro
7. Propose de recommencer ou quitter

 **Solution Complète**

* Programme : CALCULATRICE.cob

* Description : Calculatrice basique avec 4 opérations

* Auteur : Correction Exercice 3

* Date : 02/11/2024

IDENTIFICATION DIVISION.

PROGRAM-ID. CALCULATRICE.

AUTHOR. VOTRE-NOM.

DATE-WRITTEN. 02/11/2024.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

*--- Nombres pour le calcul ---

01 WS-NOMBRE1 PIC S9(5)V99.

01 WS-NOMBRE2 PIC S9(5)V99.

01 WS-RESULTAT PIC S9(7)V99.

*--- Choix de l'opération ---

01 WS-CHOIX PIC 9.

88 ADDITION VALUE 1.

88 SOUSTRACTION VALUE 2.

88 MULTIPLICATION VALUE 3.

88 DIVISION VALUE 4.

88 CHOIX-VALIDE VALUE 1 THRU 4.

*--- Contrôle du programme ---

01 WS-CONTINUER PIC X.

88 VEUT-CONTINUER VALUE "O" "o".

88 VEUT-QUITTER VALUE "N" "n".

*--- Affichage ---

01 WS-SEPARATION PIC X(60) VALUE ALL "=".

01 WS-RESULTAT-EDIT PIC -ZZZ,ZZ9.99.

PROCEDURE DIVISION.

*-----

* Programme principal - Boucle jusqu'à ce que l'utilisateur quitte

*-----

000-MAIN-PROCEDURE.

PERFORM 100-AFFICHER-BIENVENUE.

PERFORM UNTIL VEUT-QUITTER

PERFORM 200-SAISIR-NOMBRES
PERFORM 300-AFFICHER-MENU
PERFORM 400-SAISIR-CHOIX
PERFORM 500-CALCULER
PERFORM 600-AFFICHER-RESULTAT
PERFORM 700-DEMANDER-CONTINUER
END-PERFORM.

PERFORM 900-AFFICHER-AU-REVOIR.
STOP RUN.

*-----

* Afficher le message de bienvenue

*-----

100-AFFICHER-BIENVENUE.

DISPLAY " ".

DISPLAY WS-SEPARATION.

DISPLAY " CALCULATRICE COBOL ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

*-----

* Saisir les deux nombres

*-----

200-SAISIR-NOMBRES.

DISPLAY "Entrez le premier nombre : " WITH NO ADVANCING.

ACCEPT WS-NOMBRE1.

DISPLAY "Entrez le deuxieme nombre : " WITH NO ADVANCING.

ACCEPT WS-NOMBRE2.

DISPLAY " ".

*-----

* Afficher le menu des opérations

*-----

300-AFFICHER-MENU.

DISPLAY "Choisissez une operation :".

DISPLAY " 1. Addition (+)".

DISPLAY " 2. Soustraction (-)".

DISPLAY " 3. Multiplication (x)".

DISPLAY " 4. Division (/)".

DISPLAY " ".

*-----

* Saisir le choix de l'utilisateur avec validation

*-----

400-SAISIR-CHOIX.

PERFORM UNTIL CHOIX-VALIDE

DISPLAY "Votre choix (1-4) : " WITH NO ADVANCING

ACCEPT WS-CHOIX

IF NOT CHOIX-VALIDE

DISPLAY "ERREUR : Choix invalide. "

"Veuillez entrer 1, 2, 3 ou 4."

DISPLAY " "

END-IF

END-PERFORM.

DISPLAY " ".

*-----

* Effectuer le calcul selon le choix

*-----

500-CALCULER.

EVALUATE TRUE

WHEN ADDITION

ADD WS-NOMBRE1 TO WS-NOMBRE2 GIVING WS-RESULTAT

WHEN SOUSTRACTION

SUBTRACT WS-NOMBRE2 FROM WS-NOMBRE1

GIVING WS-RESULTAT

WHEN MULTIPLICATION

MULTIPLY WS-NOMBRE1 BY WS-NOMBRE2

GIVING WS-RESULTAT

WHEN DIVISION

IF WS-NOMBRE2 = ZERO

DISPLAY "ERREUR : Division par zero impossible!"

MOVE ZERO TO WS-RESULTAT

ELSE

DIVIDE WS-NOMBRE1 BY WS-NOMBRE2

GIVING WS-RESULTAT ROUNDED

END-IF

END-EVALUATE.

*-----

* Afficher le résultat du calcul

*-----

600-AFFICHER-RESULTAT.

DISPLAY WS-SEPARATION.

DISPLAY " ".

MOVE WS-RESULTAT TO WS-RESULTAT-EDIT.

EVALUATE TRUE

WHEN ADDITION

DISPLAY WS-NOMBRE1 " + " WS-NOMBRE2

" = " WS-RESULTAT-EDIT

WHEN SOUSTRATION

DISPLAY WS-NOMBRE1 " - " WS-NOMBRE2

" = " WS-RESULTAT-EDIT

WHEN MULTIPLICATION

DISPLAY WS-NOMBRE1 " x " WS-NOMBRE2

" = " WS-RESULTAT-EDIT

WHEN DIVISION

IF WS-NOMBRE2 NOT = ZERO

DISPLAY WS-NOMBRE1 " / " WS-NOMBRE2

" = " WS-RESULTAT-EDIT

END-IF

END-EVALUATE.

DISPLAY " ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

*-----

* Demander si l'utilisateur veut continuer

*-----

700-DEMANDER-CONTINUER.

DISPLAY "Voulez-vous faire un autre calcul ? (O/N) : "

WITH NO ADVANCING.

ACCEPT WS-CONTINUER.

DISPLAY " ".

*-----

* Afficher le message d'au revoir

*-----

900-AFFICHER-AU-REVOIR.

DISPLAY WS-SEPARATION.

DISPLAY " Merci d'avoir utilise la calculatrice! ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

Explications des Concepts Avancés

1. Boucle PERFORM UNTIL

PERFORM UNTIL VEUT-QUITTER

PERFORM 200-SAISIR-NOMBRES

...

END-PERFORM.

- Continue tant que VEUT-QUITTER est FALSE
- VEUT-QUITTER est un niveau 88 qui teste si WS-CONTINUER = "N"

2. Validation avec Boucle

PERFORM UNTIL CHOIX-VALIDE

ACCEPT WS-CHOIX

IF NOT CHOIX-VALIDE

DISPLAY "Erreur..."

END-IF

END-PERFORM.

- Ne sort de la boucle que si choix entre 1 et 4
- CHOIX-VALIDE = niveau 88 avec VALUE 1 THRU 4

3. EVALUATE (comme switch/case)

EVALUATE TRUE

WHEN ADDITION

ADD ...

WHEN SOUSTRACTION

SUBTRACT ...

...

END-EVALUATE.

- Plus lisible qu'une série de IF imbriqués
- Utilise les niveaux 88 comme conditions

4. Protection Division par Zéro

IF WS-NOMBRE2 = ZERO

DISPLAY "ERREUR : Division par zero impossible!"

```

        MOVE ZERO TO WS-RESULTAT
    ELSE
        DIVIDE WS-NOMBRE1 BY WS-NOMBRE2 GIVING WS-RESULTAT ROUNDED
    END-IF.

```

- **Toujours** tester avant de diviser
- ROUNDED : Arrondit le résultat

5. Format d'Affichage Édité

```
01 WS-RESULTAT-EDIT PIC -ZZZ,ZZ9.99.
```

- - : Affiche le signe négatif si besoin
- Z : Remplace les zéros non significatifs par des espaces
- , : Séparateur de milliers
- 9 : Chiffre toujours affiché
- .99 : Deux décimales toujours affichées

Exemple :

WS-RESULTAT = 12.50 → WS-RESULTAT-EDIT = " 12.50"

WS-RESULTAT = 1234.56 → WS-RESULTAT-EDIT = " 1,234.56"

WS-RESULTAT = -50.00 → WS-RESULTAT-EDIT = " -50.00"

Exemple d'Exécution

```

=====
          CALCULATRICE COBOL
=====

```

Entrez le premier nombre : 25.5

Entrez le deuxieme nombre : 10

Choisissez une operation :

1. Addition (+)
2. Soustraction (-)

3. Multiplication (x)

4. Division (/)

Votre choix (1-4) : 1

=====

25.50 + 10.00 = 35.50

=====

Voulez-vous faire un autre calcul ? (O/N) : O

Entrez le premier nombre : 100

Entrez le deuxieme nombre : 0

Choisissez une operation :

1. Addition (+)

2. Soustraction (-)

3. Multiplication (x)

4. Division (/)

Votre choix (1-4) : 4

ERREUR : Division par zero impossible!

=====

100.00 / 0.00 = (erreur)

=====

Voulez-vous faire un autre calcul ? (O/N) : N

=====

Merci d'avoir utilise la calculatrice!

=====

Améliorations Possibles

Amélioration 1 : Historique des Calculs

01 WS-HISTORIQUE OCCURS 10 TIMES.

05 HIST-OPERATION PIC X(50).

Amélioration 2 : Plus d'Opérations

01 WS-CHOIX PIC 9.

88 PUISSANCE VALUE 5.

88 RACINE-CARREE VALUE 6.

88 MODULO VALUE 7.

Amélioration 3 : Sauvegarder dans un Fichier

FILE SECTION.

FD HISTORIQUE-FILE.

01 HIST-ENREG PIC X(100).

Pièges à Éviter

1. Ne pas valider le choix

 ACCEPT WS-CHOIX.

EVALUATE WS-CHOIX ... *> Peut être n'importe quoi!

 PERFORM UNTIL CHOIX-VALIDE

ACCEPT WS-CHOIX

...

END-PERFORM.

2. Oublier la division par zéro

✗ DIVIDE WS-NOMBRE1 BY WS-NOMBRE2 GIVING WS-RESULTAT.

*> CRASH si WS-NOMBRE2 = 0 !

✓ IF WS-NOMBRE2 = ZERO

DISPLAY "Erreur"

ELSE

DIVIDE ...

END-IF.

3. Mauvais format pour nombres négatifs

✗ 01 WS-NOMBRE PIC 9(5)V99. *> Pas de signe!

✓ 01 WS-NOMBRE PIC S9(5)V99. *> Avec signe

🎓 Points Clés à Retenir

1. ✓ **PERFORM UNTIL** : Boucle conditionnelle
2. ✓ **EVALUATE TRUE** : Alternative élégante aux IF multiples
3. ✓ **Niveau 88** : Rend le code très lisible
4. ✓ **Validation d'entrée** : Toujours valider avant de traiter
5. ✓ **Division par zéro** : TOUJOURS tester avant
6. ✓ **PICTURE éditées** : Pour un affichage professionnel
7. ✓ **ROUNDED** : Arrondir les divisions

Partie 2 : Programmation de Base

Exercice 4 : Opérations Arithmétiques

📄 Énoncé

Crée un programme qui calcule les statistiques d'une série de notes :

1. Demande combien de notes (maximum 20)
2. Saisie de chaque note (0-20)
3. Calcule et affiche :
 - La somme totale
 - La moyenne
 - La note la plus haute
 - La note la plus basse
 - Le nombre de notes ≥ 10 (admis)
 - Le pourcentage de réussite

 **Solution Complète**

* Programme : STATISTIQUES-NOTES.cob

* Description : Calcule les statistiques d'une série de notes

* Auteur : Correction Exercice 4

* Date : 02/11/2024

IDENTIFICATION DIVISION.

PROGRAM-ID. STATS-NOTES.

DATA DIVISION.

WORKING-STORAGE SECTION.

*--- Compteurs et limites ---

01 WS-NB-NOTES PIC 99 VALUE ZERO.

01 WS-LIMITE-MAX PIC 99 VALUE 20.

01 WS-COMPTEUR PIC 99 VALUE ZERO.

01 WS-NB-ADMIS PIC 99 VALUE ZERO.

*--- Notes et calculs ---

01 WS-NOTE-ACTUELLE PIC 99V99.

01 WS-SOMME-NOTES PIC 9(5)V99 VALUE ZERO.

01 WS-MOYENNE PIC 99V99.

01 WS-NOTE-MAX PIC 99V99 VALUE ZERO.

01 WS-NOTE-MIN PIC 99V99 VALUE 20.

01 WS-POURCENTAGE PIC 999V99.

*--- Affichage ---

01 WS-SEPARATION PIC X(60) VALUE ALL "=".

01 WS-TIRETS PIC X(60) VALUE ALL "-".

01 WS-MOYENNE-EDIT PIC Z9.99.

01 WS-POURCENTAGE-EDIT PIC ZZ9.99.

*--- Validation ---

01 WS-NOTE-VALIDE-FLAG PIC X.

88 NOTE-VALIDE VALUE "O".

88 NOTE-INVALIDE VALUE "N".

PROCEDURE DIVISION.

000-MAIN-PROCEDURE.

PERFORM 100-INITIALISATION

PERFORM 200-SAISIR-NB-NOTES

PERFORM 300-SAISIR-NOTES

PERFORM 400-CALCULER-STATISTIQUES

PERFORM 500-AFFICHER-RESULTATS

STOP RUN.

*-----

* Afficher l'en-tête

*-----

100-INITIALISATION.

DISPLAY " ".

DISPLAY WS-SEPARATION.

DISPLAY " STATISTIQUES DE NOTES (sur 20)".

DISPLAY WS-SEPARATION.

DISPLAY " ".

*-----

* Demander le nombre de notes avec validation

*-----

200-SAISIR-NB-NOTES.

PERFORM UNTIL WS-NB-NOTES > 0 AND WS-NB-NOTES <= WS-LIMITE-MAX

DISPLAY "Combien de notes voulez-vous entrer (1-20) ? "

WITH NO ADVANCING

ACCEPT WS-NB-NOTES

IF WS-NB-NOTES <= 0 OR WS-NB-NOTES > WS-LIMITE-MAX

DISPLAY "ERREUR : Entrez un nombre entre 1 et 20."

DISPLAY " "

END-IF

END-PERFORM.

DISPLAY " ".

*-----

* Saisir toutes les notes avec validation

*-----

300-SAISIR-NOTES.

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1

UNTIL WS-COMPTEUR > WS-NB-NOTES

SET NOTE-INVALIDE TO TRUE

PERFORM UNTIL NOTE-VALIDE

DISPLAY "Note " WS-COMPTEUR " (0-20) : "

WITH NO ADVANCING

ACCEPT WS-NOTE-ACTUELLE

IF WS-NOTE-ACTUELLE >= 0 AND WS-NOTE-ACTUELLE <= 20

SET NOTE-VALIDE TO TRUE

PERFORM 310-TRAITER-NOTE

ELSE

DISPLAY " ERREUR : La note doit etre entre 0 et 20."

END-IF

END-PERFORM

END-PERFORM.

DISPLAY " ".

*-----

* Traiter une note individuelle

*-----

310-TRAITER-NOTE.

*--- Ajouter à la somme ---

ADD WS-NOTE-ACTUELLE TO WS-SOMME-NOTES.

*--- Mettre à jour le maximum ---

IF WS-NOTE-ACTUELLE > WS-NOTE-MAX

MOVE WS-NOTE-ACTUELLE TO WS-NOTE-MAX

END-IF.

*--- Mettre à jour le minimum ---

IF WS-NOTE-ACTUELLE < WS-NOTE-MIN

MOVE WS-NOTE-ACTUELLE TO WS-NOTE-MIN

END-IF.

*--- Compter les admis (≥ 10) ---

IF WS-NOTE-ACTUELLE ≥ 10

ADD 1 TO WS-NB-ADMIS

END-IF.

*-----

* Calculer la moyenne et le pourcentage

*-----

400-CALCULER-STATISTIQUES.

*--- Calculer la moyenne ---

DIVIDE WS-SOMME-NOTES BY WS-NB-NOTES

GIVING WS-MOYENNE ROUNDED.

*--- Calculer le pourcentage de réussite ---

COMPUTE WS-POURCENTAGE ROUNDED =
(WS-NB-ADMIS / WS-NB-NOTES) * 100.

*-----

* Afficher tous les résultats

*-----

500-AFFICHER-RESULTATS.

DISPLAY WS-SEPARATION.

DISPLAY " RESULTATS ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

DISPLAY "Nombre de notes : " WS-NB-NOTES.

DISPLAY WS-TIRETS.

DISPLAY "Somme totale : " WS-SOMME-NOTES.

MOVE WS-MOYENNE TO WS-MOYENNE-EDIT.

DISPLAY "Moyenne : " WS-MOYENNE-EDIT " / 20".

DISPLAY WS-TIRETS.

DISPLAY "Note la plus haute : " WS-NOTE-MAX.

DISPLAY "Note la plus basse : " WS-NOTE-MIN.

DISPLAY WS-TIRETS.

DISPLAY "Nombre d'admis (>=10): " WS-NB-ADMIS.

MOVE WS-POURCENTAGE TO WS-POURCENTAGE-EDIT.

DISPLAY "Taux de reussite : " WS-POURCENTAGE-EDIT " %".

DISPLAY " ".

DISPLAY WS-SEPARATION.

Explications des Concepts

1. Boucle **PERFORM VARYING** (équivalent **FOR**)

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1

UNTIL WS-COMPTEUR > WS-NB-NOTES

...

END-PERFORM.

- FROM 1 : Commence à 1
- BY 1 : Incrémente de 1 à chaque tour
- UNTIL : Condition d'arrêt

Équivalent en Python :

for compteur in range(1, nb_notes + 1):

...

2. Validation Imbriquée

PERFORM UNTIL condition-externe

SET NOTE-INVALIDE TO TRUE

PERFORM UNTIL NOTE-VALIDE

ACCEPT ...

IF valide

SET NOTE-VALIDE TO TRUE

END-IF

END-PERFORM

END-PERFORM.

- Boucle externe : Chaque note
- Boucle interne : Validation de la note

3. Recherche de Min/Max

*--- Maximum ---

IF WS-NOTE-ACTUELLE > WS-NOTE-MAX

MOVE WS-NOTE-ACTUELLE TO WS-NOTE-MAX

END-IF.

*--- Minimum ---

IF WS-NOTE-ACTUELLE < WS-NOTE-MIN

MOVE WS-NOTE-ACTUELLE TO WS-NOTE-MIN

END-IF.

- Initialiser max à 0 et min à 20
- Comparer chaque note

4. Calcul de Pourcentage

COMPUTE WS-POURCENTAGE ROUNDED =

(WS-NB-ADMIS / WS-NB-NOTES) * 100.

- Parenthèses = priorité des opérations
- ROUNDED = Arrondi au plus proche
- * 100 = Convertit en pourcentage

Exemple d'Exécution

=====

STATISTIQUES DE NOTES (sur 20)

=====

Combien de notes voulez-vous entrer (1-20) ? 5

Note 1 (0-20) : 15.5

Note 2 (0-20) : 12

Note 3 (0-20) : 8.5

Note 4 (0-20) : 16

Note 5 (0-20) : 25

ERREUR : La note doit etre entre 0 et 20.

Note 5 (0-20) : 14

=====

RESULTATS

=====

Nombre de notes : 05

Somme totale : 66.00

Moyenne : 13.20 / 20

Note la plus haute : 16.00

Note la plus basse : 08.50

Nombre d'admis (≥ 10): 04

Taux de reussite : 80.00 %

=====

Améliorations Possibles

Amélioration 1 : Stocker toutes les notes dans un tableau

01 WS-TABLEAU-NOTES OCCURS 20 TIMES.

05 NOTE PIC 99V99.

Amélioration 2 : Calculer l'écart-type

01 WS-ECART-TYPE PIC 99V99.

COMPUTE WS-ECART-TYPE = FUNCTION STANDARD-DEVIATION(...)

Amélioration 3 : Afficher un histogramme

DISPLAY "Distribution :".

DISPLAY "[00-05] : " WS-TRANCHE1 " notes".

DISPLAY "[06-09] : " WS-TRANCHE2 " notes".

DISPLAY "[10-14] : " WS-TRANCHE3 " notes".

DISPLAY "[15-20] : " WS-TRANCHE4 " notes".

⚠ Pièges à Éviter

1. Division par zéro

✗ DIVIDE WS-SOMME BY WS-NB-NOTES GIVING WS-MOYENNE.

*> Si WS-NB-NOTES = 0 → CRASH!

✓ IF WS-NB-NOTES > 0

DIVIDE WS-SOMME BY WS-NB-NOTES GIVING WS-MOYENNE

ELSE

MOVE ZERO TO WS-MOYENNE

END-IF.

2. Oublier d'initialiser min/max

✗ 01 WS-NOTE-MAX PIC 99V99. *> Initialisé à 0 par défaut

01 WS-NOTE-MIN PIC 99V99. *> Initialisé à 0 par défaut

*> Problème : Toutes les notes > 0, donc max sera OK

*> Mais min restera à 0!

✓ 01 WS-NOTE-MAX PIC 99V99 VALUE ZERO.

01 WS-NOTE-MIN PIC 99V99 VALUE 20.

3. Débordement de la somme

✗ 01 WS-SOMME PIC 999V99. *> Max = 999.99

*> 20 notes de 20 = 400 → OK

*> Mais si on change le programme...

✓ 01 WS-SOMME PIC 9(5)V99. *> Max = 99999.99

🎓 Points Clés à Retenir

1. ✓ **PERFORM VARYING** : Boucle compteur (comme FOR)
 2. ✓ **Validation imbriquée** : Boucles dans boucles
 3. ✓ **Min/Max** : Comparer et mettre à jour
 4. ✓ **COMPUTE** : Formules mathématiques complexes
 5. ✓ **ROUNDED** : Arrondir les divisions
 6. ✓ **Initialisation** : Toujours initialiser min/max correctement
-

Exercice 5 : Manipulation de Chaînes

📄 Énoncé

Crée un programme qui :

1. Demande une phrase
2. Affiche :
 - La phrase en majuscules
 - La phrase en minuscules
 - La phrase inversée
 - Le nombre de mots
 - Le nombre de voyelles
 - Le nombre de consonnes
3. Demande un mot à rechercher et affiche combien de fois il apparaît

✓ Solution Complète

* Programme : ANALYSE-TEXTE.cob

* Description : Analyse complète d'une phrase

* Auteur : Correction Exercice 5

* Date : 02/11/2024

IDENTIFICATION DIVISION.

PROGRAM-ID. ANALYSE-TEXTE.

DATA DIVISION.

WORKING-STORAGE SECTION.

*--- Chaînes de caractères ---

01 WS-PHRASE PIC X(200).

01 WS-PHRASE-MAJ PIC X(200).

01 WS-PHRASE-MIN PIC X(200).

01 WS-PHRASE-INVERSE PIC X(200).

01 WS-MOT-RECHERCHE PIC X(30).

*--- Compteurs ---

01 WS-NB-MOTS PIC 999 VALUE ZERO.

01 WS-NB-VOYELLES PIC 999 VALUE ZERO.

01 WS-NB-CONSONNES PIC 999 VALUE ZERO.

01 WS-NB-OCCURRENCES PIC 999 VALUE ZERO.

01 WS-LONGUEUR PIC 999.

01 WS-COMPTEUR PIC 999.

*--- Variables de travail ---

01 WS-CARACTERE PIC X.

01 WS-PRECEDENT PIC X VALUE SPACE.

*--- Affichage ---

01 WS-SEPARATION PIC X(70) VALUE ALL "=".

01 WS-TIRETS PIC X(70) VALUE ALL "-".

PROCEDURE DIVISION.

000-MAIN-PROCEDURE.

PERFORM 100-AFFICHER-ENTETE

PERFORM 200-SAISIR-PHRASE

PERFORM 300-CONVERTIR-MAJUSCULES

PERFORM 310-CONVERTIR-MINUSCULES

PERFORM 320-INVERSER-PHRASE

PERFORM 400-COMPTER-MOTS

PERFORM 500-COMPTER-VOYELLES-CONSONNES

PERFORM 600-AFFICHER-RESULTATS

PERFORM 700-RECHERCHER-MOT

STOP RUN.

*-----

* En-tête du programme

*-----

100-AFFICHER-ENTETE.

DISPLAY " ".

DISPLAY WS-SEPARATION.

DISPLAY " ANALYSEUR DE TEXTE ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

*-----

* Saisir la phrase

*-----

200-SAISIR-PHRASE.

DISPLAY "Entrez une phrase (max 200 caracteres) :".

ACCEPT WS-PHRASE.

DISPLAY " ".

*-----

* Convertir en majuscules

*-----

300-CONVERTIR-MAJUSCULES.

MOVE FUNCTION UPPER-CASE(WS-PHRASE) TO WS-PHRASE-MAJ.

*-----

* Convertir en minuscules

*-----

310-CONVERTIR-MINUSCULES.

MOVE FUNCTION LOWER-CASE(WS-PHRASE) TO WS-PHRASE-MIN.

*-----

* Inverser la phrase

*-----

320-INVERSER-PHRASE.

MOVE FUNCTION REVERSE(WS-PHRASE) TO WS-PHRASE-INVERSE.

*-----

* Compter le nombre de mots

*-----

400-COMPTER-MOTS.

MOVE ZERO TO WS-NB-MOTS.

MOVE SPACE TO WS-PRECEDENT.

COMPUTE WS-LONGUEUR = FUNCTION LENGTH(
FUNCTION TRIM(WS-PHRASE TRAILING)
).

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1
UNTIL WS-COMPTEUR > WS-LONGUEUR

MOVE WS-PHRASE(WS-COMPTEUR:1) TO WS-CARACTERE

IF WS-PRECEDENT = SPACE AND WS-CARACTERE NOT = SPACE
ADD 1 TO WS-NB-MOTS
END-IF

MOVE WS-CARACTERE TO WS-PRECEDENT
END-PERFORM.

*-----

* Compter voyelles et consonnes

*-----

500-COMPTER-VOYELLES-CONSONNES.

MOVE ZERO TO WS-NB-VOYELLES.

MOVE ZERO TO WS-NB-CONSONNES.

COMPUTE WS-LONGUEUR = FUNCTION LENGTH(WS-PHRASE).

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1

UNTIL WS-COMPTEUR > WS-LONGUEUR

MOVE WS-PHRASE(WS-COMPTEUR:1) TO WS-CARACTERE

EVALUATE WS-CARACTERE

WHEN "A"

WHEN "E"

WHEN "I"

WHEN "O"

WHEN "U"

WHEN "Y"

WHEN "a"

WHEN "e"

WHEN "i"

WHEN "o"

WHEN "u"

WHEN "y"

ADD 1 TO WS-NB-VOYELLES

WHEN "B" THRU "Z"

WHEN "b" THRU "z"

ADD 1 TO WS-NB-CONSONNES

END-EVALUATE

END-PERFORM.

*-----

* Afficher tous les résultats

*-----

600-AFFICHER-RESULTATS.

DISPLAY WS-SEPARATION.

DISPLAY " RESULTATS DE L'ANALYSE ".

DISPLAY WS-SEPARATION.

DISPLAY " ".

DISPLAY "Phrase originale :".

DISPLAY " " WS-PHRASE.

DISPLAY " ".

DISPLAY WS-TIRETS.

DISPLAY "En MAJUSCULES :".

DISPLAY " " WS-PHRASE-MAJ.

DISPLAY " ".

DISPLAY "En minuscules :".

DISPLAY " " WS-PHRASE-MIN.

DISPLAY " ".

DISPLAY "Inversee :".

DISPLAY " " WS-PHRASE-INVERSE.

DISPLAY " ".

DISPLAY WS-TIRETS.

DISPLAY "Statistiques :".

DISPLAY " Nombre de mots : " WS-NB-MOTS.

DISPLAY " Nombre de voyelles : " WS-NB-VOYELLES.

DISPLAY " Nombre de consonnes : " WS-NB-CONSONNES.

DISPLAY " ".

*-----

* Rechercher un mot dans la phrase

*-----

700-RECHERCHER-MOT.

DISPLAY WS-SEPARATION.

DISPLAY "Entrez un mot a rechercher : " WITH NO ADVANCING.

ACCEPT WS-MOT-RECHERCHE.

MOVE ZERO TO WS-NB-OCCURRENCES.

INSPECT WS-PHRASE TALLYING WS-NB-OCCURRENCES
FOR ALL WS-MOT-RECHERCHE.

DISPLAY " ".

DISPLAY "Le mot " WS-MOT-RECHERCHE " apparait "

WS-NB-OCCURRENCES " fois."

DISPLAY " ".

DISPLAY WS-SEPARATION.

Explications Détaillées

1. Fonctions de Chaînes

FUNCTION UPPER-CASE(chaîne) → Majuscules

FUNCTION LOWER-CASE(chaîne) → Minuscules

FUNCTION REVERSE(chaîne) → Inversion

FUNCTION LENGTH(chaîne) → Longueur

FUNCTION TRIM(chaîne TRAILING) → Sans espaces de fin

2. Référence Modification (Extraction)

WS-PHRASE(position:longueur)

Exemples :

WS-PHRASE(1:1) → Premier caractère

WS-PHRASE(5:3) → 3 caractères à partir de la position 5

WS-PHRASE(WS-COMPTEUR:1) → Caractère à la position WS-COMPTEUR

3. Compter les Mots

MOVE SPACE TO WS-PRECEDENT.

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1

UNTIL WS-COMPTEUR > WS-LONGUEUR

MOVE WS-PHRASE(WS-COMPTEUR:1) TO WS-CARACTERE

IF WS-PRECEDENT = SPACE AND WS-CARACTERE NOT = SPACE


ADD 1 TO WS-NB-MOTS

END-IF

MOVE WS-CARACTERE TO WS-PRECEDENT

END-PERFORM.

Logique :

- On compte une transition espace → non-espace
- Exemple : "Hello World"
 - Position 1 : SPACE → 'H' = 1 mot 
 - Position 7 : 'o' → SPACE = pas un nouveau mot

- Position 8 : SPACE → 'W' = 2 mots 

4. EVALUATE avec THRU (plages)

EVALUATE WS-CARACTERE

WHEN "A"

WHEN "E"

WHEN "I"

ADD 1 TO WS-NB-VOYELLES

WHEN "B" THRU "Z" *> De B à Z (toutes les consonnes)

ADD 1 TO WS-NB-CONSONNES

END-EVALUATE.

5. INSPECT TALLYING (Compter occurrences)

INSPECT WS-PHRASE TALLYING WS-NB-OCCURRENCES

FOR ALL WS-MOT-RECHERCHE.

- Compte combien de fois WS-MOT-RECHERCHE apparaît dans WS-PHRASE
- FOR ALL = Toutes les occurrences

Exemple d'Exécution

=====

ANALYSEUR DE TEXTE

=====

Entrez une phrase (max 200 caracteres) :

Bonjour le monde COBOL est un langage puissant

=====

RESULTATS DE L'ANALYSE

=====

Phrase originale :

Bonjour le monde COBOL est un langage puissant

En MAJUSCULES :

BONJOUR LE MONDE COBOL EST UN LANGAGE PUISSANT

En minuscules :

bonjour le monde cobol est un langage puissant

Inversée :

tnassiup egagnal nu tse LOBOC ednom el ruojnoB

Statistiques :

Nombre de mots : 008

Nombre de voyelles : 016

Nombre de consonnes : 023

=====

Entrez un mot à rechercher : un

Le mot 'un' apparaît 001 fois.

=====

💡 **Améliorations Possibles**

Amélioration 1 : Recherche Insensible à la Casse

MOVE FUNCTION UPPER-CASE(WS-PHRASE) TO WS-PHRASE-MAJ.

MOVE FUNCTION UPPER-CASE(WS-MOT-RECHERCHE) TO WS-MOT-MAJ.

INSPECT WS-PHRASE-MAJ TALLYING WS-NB-OCCURRENCES
FOR ALL WS-MOT-MAJ.

Amélioration 2 : Compter les Chiffres

01 WS-NB-CHIFFRES PIC 999 VALUE ZERO.


EVALUATE WS-CARACTERE
WHEN "0" THRU "9"
ADD 1 TO WS-NB-CHIFFRES
END-EVALUATE.

Amélioration 3 : Supprimer Espaces Multiples

INSPECT WS-PHRASE REPLACING ALL " " BY " ".

Pièges à Éviter

1. Dépasser la longueur de la chaîne

 MOVE WS-PHRASE(250:1) TO WS-CARACTERE.

*> Si WS-PHRASE fait 200 caractères → ERREUR!

 COMPUTE WS-LONGUEUR = FUNCTION LENGTH(WS-PHRASE).

PERFORM VARYING WS-COMPTEUR FROM 1 BY 1

UNTIL WS-COMPTEUR > WS-LONGUEUR

...

END-PERFORM.

2. Oublier les minuscules dans EVALUATE

 EVALUATE WS-CARACTERE

WHEN "A"

WHEN "E" *> Oublie "a" et "e" !

✅ EVALUATE WS-CARACTERE

WHEN "A"

WHEN "E"

WHEN "a" *> Inclure minuscules

WHEN "e"

3. **INSPECT case-sensitive**

WS-PHRASE = "Hello hello HELLO"

❌ INSPECT WS-PHRASE TALLYING WS-COUNT FOR ALL "hello".

*> Compte seulement 1 ("hello" en minuscules)

✅ MOVE FUNCTION UPPER-CASE(WS-PHRASE) TO WS-TEMP.

INSPECT WS-TEMP TALLYING WS-COUNT FOR ALL "HELLO".

*> Compte 3

🎓 **Points Clés à Retenir**


1. ✅ **Fonctions COBOL** : UPPER-CASE, LOWER-CASE, REVERSE, LENGTH
2. ✅ **Référence modification** : chaine(pos:long)
3. ✅ **INSPECT TALLYING** : Compter des occurrences
4. ✅ **EVALUATE ... THRU** : Plages de valeurs
5. ✅ **Compter les mots** : Transition espace → non-espace
6. ✅ **Parcourir une chaîne** : Boucle caractère par caractère

🚀 **Suite dans le prochain fichier...**

Cette correction couvre les 5 premiers exercices en détail.

À venir :

- ✅ Exercices 6-10 (Structures conditionnelles, boucles, fichiers)
- ✅ Exercices avancés (Tables, SORT, sous-programmes)

-  Projets complets (Système bancaire, Gestion stock, Paie)

Temps d'étude estimé pour cette partie : 15-20 heures

Continue vers la suite des corrections !  