📚 COBOL : Guide Complet du Débutant à l'Expert - Partie 1

🎯 Bienvenue dans le Monde du COBOL !

"COBOL n'est pas mort - il fait tourner 95% des distributeurs bancaires et 80% des transactions financières mondiales !"

📋 Table des Matières - Partie 1

---

1. Introduction au COBOL

1.1 Qu'est-ce que le COBOL ?

COBOL signifie **CO**mmon **B**usiness-**O**riented **L**anguage

C'est un langage de programmation créé en **1959** (oui, il a plus de 65 ans !) et qui est TOUJOURS massivement utilisé aujourd'hui.

🤔 Pourquoi COBOL Existe Encore ?

Raisons principales :

1. **Legacy Systems (Systèmes Hérités)**
   - Des millions de lignes de code COBOL en production
   - Banques, assurances, gouvernements
   - Trop coûteux/risqué de tout réécrire
2. **Fiabilité Éprouvée**
   - 65+ ans de stabilité
   - Testé et éprouvé

- Moins de bugs que les langages modernes

3. **Transactions Financières**
   - 95% des distributeurs automatiques
   - 80% des transactions en personne
   - Systèmes de paie de grandes entreprises
4. **Données Massives**
   - Traitement de fichiers gigantesques
   - Optimisé pour les opérations batch
   - Gère des milliards d'enregistrements

💰 Le Marché du COBOL en 2024

Aspect	Statistique
Lignes de code en production	220+ milliards
Transactions quotidiennes	3 milliards
Salaire moyen USA	$75,000 - $120,000
Demande	⬆️ En hausse (pénurie de compétences)
Âge moyen des développeurs COBOL	55+ ans (besoin de jeunes !)

🎯 Qui Utilise COBOL Aujourd'hui ?

Secteurs principaux :

1. 🏦 **Banques et Finance**
   - JPMorgan Chase
   - Bank of America
   - Wells Fargo
   - Crédit Agricole
2. 📮 **Assurances**
   - AXA
   - Allianz
   - MetLife

3. 🏛️ **Gouvernements**
    - IRS (impôts USA)
    - Social Security Administration
    - DMV (permis de conduire)
    - Sécurité Sociale (France)
4. ✈️ **Transport**
    - Systèmes de réservation aérienne
    - Sabre (American Airlines)
5. 🏢 **Grandes Entreprises**
    - IBM
    - Walmart
    - FedEx

1.2 Pourquoi Apprendre COBOL en 2024 ?

✅ **Avantages d'Apprendre COBOL**

1. Opportunités d'Emploi

📊 Situation actuelle :

- Développeurs COBOL : < 50,000 aux USA

- Postes disponibles : > 100,000

- Ratio : 2 postes pour 1 développeur !

2. Salaires Élevés

💰 Fourchettes de salaires (USA) :

Junior (0-2 ans) : $55,000 - $75,000

Intermédiaire (3-5) : $75,000 - $95,000

Senior (5-10) : $95,000 - $120,000

Expert (10+ ans) : $120,000 - $180,000+

En freelance : $100 - $200/heure

3. Stabilité de Carrière

🔒 Avantages :

- Peu de concurrence

- Projets critiques

- Impossible à délocaliser facilement

- Connaissance rare et précieuse

4. Modernisation en Cours

🚀 Tendances :

- Migration vers le cloud (AWS, Azure)

- Conteneurisation (Docker)

- APIs RESTful depuis COBOL

- Interfaces modernes sur systèmes anciens

❌ **Défis à Considérer**

1. **Perception Dépassée**
    - Considéré comme "vieux"
    - Moins sexy que Python/JavaScript
    - Communauté plus petite
2. **Apprentissage Initial**
    - Syntaxe très différente des langages modernes
    - Concepts d'une autre époque
    - Moins de ressources en ligne
3. **Environnement**
    - Mainframe complexes
    - Outils moins modernes
    - Documentation parfois obsolète

1.3 COBOL vs Langages Modernes

📊 **Comparaison Détaillée**

Aspect	COBOL	Python	Java
Année de création	1959	1991	1995
Paradigme	Procédural	Multi-paradigme	Orienté objet
Syntaxe	Verbeux (mots anglais)	Concis	Modéré
Courbe d'apprentissage	Moyenne	Facile	Difficile
Performance	Excellente (batch)	Moyenne	Bonne
Écosystème	Limité	Énorme	Très grand
Salaire moyen	$90K	$80K	$85K
Demande	Haute (niche)	Très haute	Très haute

💡 **Exemple de Code Comparatif**

Afficher "Hello World"

COBOL :

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID. HELLO.
    PROCEDURE DIVISION.
      DISPLAY "Hello World".
      STOP RUN.
```

Python :

```
print("Hello World")
```

Java :

```
public class Hello {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Analyse :

- COBOL : Verbeux mais TRÈS lisible

- Python : Ultra concis
- Java : Structure orientée objet

1.4 Ce Que Tu Vas Apprendre

📚 **Partie 1 : Fondamentaux (Tu es ici !)**

- ✅ Introduction et contexte
- ✅ Installation de l'environnement
- ✅ Structure des programmes
- ✅ Types de données
- ✅ Variables et PICTURE clauses

📗 **Partie 2 : Programmation de Base**

- Opérations arithmétiques
- Structures conditionnelles (IF)
- Boucles (PERFORM)
- Manipulation de chaînes
- Exercices pratiques

📘 **Partie 3 : Fichiers et Données**

- Types de fichiers (Sequential, Indexed, Relative)
- Lecture et écriture
- COPY et bibliothèques
- Sort et Merge
- Projets pratiques

📕 **Partie 4 : Avancé**

- Tables et tableaux
- Sous-programmes
- SQL embarqué (DB2)
- CICS et transactions
- JCL (Job Control Language)

📕 Partie 5 : Expert

- Optimisation de performance
- Debugging avancé
- Modernisation (APIs REST)
- Migration vers le cloud
- Projet final complet

1.5 Comment Utiliser Ce Guide

🎯 Approche Recommandée

Pour les Débutants Complets :

1. Lis chaque section dans l'ordre
2. Ne saute AUCUNE explication
3. Tape TOUS les exemples de code
4. Fais TOUS les exercices
5. Prends des notes manuscrites
6. Pratique 30 min/jour minimum

Pour ceux avec Expérience en Programmation :

1. Lis les sections théoriques en diagonale
2. Concentre-toi sur les exemples
3. Fais les exercices avancés
4. Compare avec tes langages connus
5. Pratique 1h/jour pour accélérer

⏰ Durée Estimée

Niveau	Durée Totale	Heures/Semaine	Semaines
Débutant	80-100h	10-15h	6-10
Avec Expérience	50-70h	10-15h	4-7
Mode Intensif	40-60h	20-30h	2-3

📝 Notation des Exercices

Symboles utilisés :

- ⭐ : Facile (essentiel)
- ⭐ ⭐ : Moyen (recommandé)
- ⭐ ⭐ ⭐ : Difficile (challenge)
- 🔥 : Projet complet
- 💡 : Astuce importante
- ⚠️ : Attention/Piège courant

---

2. Histoire et Contexte du COBOL

2.1 La Naissance du COBOL (1959)

🏛️ Contexte Historique

1950s - Le Problème :

Situation :

- Chaque ordinateur avait son propre langage
- Pas de standardisation
- Code non portable
- Très difficile d'échanger des programmes

Exemple :

IBM avait son langage, UNIVAC un autre, etc.

La Solution : COBOL

Objectif : Créer un langage COMMUN pour le BUSINESS

- Portable entre différentes machines
- Lisible par les non-programmeurs
- Orienté traitement de données commerciales

👥 Les Créateurs

Grace Hopper (1906-1992) 🌟

Surnommée : "Amazing Grace"

Rôle : Inventrice des compilateurs

Contributions :

- Premier compilateur (A-0, 1952)
- Idée d'un langage en anglais
- Membre clé du comité COBOL
- Amiral de la US Navy

Citation célèbre :

"Il est plus facile de demander pardon que la permission"

Le Comité CODASYL

CODASYL = Conference on Data Systems Languages

Membres :

- Gouvernement US (Department of Defense)
- Constructeurs (IBM, UNIVAC, Honeywell)
- Utilisateurs (banques, assurances)

Date : Mai 1959

Lieu : Pentagone, Washington DC

2.2 Évolution du COBOL

🗓️ **Timeline des Versions**

1960 : COBOL-60

Première version officielle

- Syntaxe de base établie
- 4 divisions
- Fichiers séquentiels

1968 : COBOL-68

Standardisation ANSI

- Sort interne

- Report Writer

- Amélioration des fichiers

1974 : COBOL-74

Améliorations majeures

- Nested programs

- INSPECT statement

- Fichiers indexed

1985 : COBOL-85

Modernisation

- Structured programming

- EVALUATE (comme switch)

- Inline PERFORM

- INITIALIZE

2002 : COBOL 2002

Entrée dans le 21ème siècle

- Support Unicode

- Objects (OO COBOL)

- Locale

- User-defined functions

2014 : COBOL 2014

Version actuelle

- Dynamic tables

- XML/JSON support

- Meilleur support des pointeurs

📈 **Impact Historique**

Années 60-70 : L'Âge d'Or

📊 Statistiques :

- 1970 : 80% des applications business en COBOL

- Milliers de programmeurs formés

- Standard de l'industrie

Raisons :

✅ Portable

✅ Lisible

✅ Soutenu par le gouvernement

✅ Excellente documentation

Années 80-90 : Consolidation

📊 Développements :

- Migration vers PC

- Systèmes client-serveur

- COBOL reste dominant pour mainframe

Problème Y2K (2000) :

💰 $300+ milliards dépensés

👨‍💻 Pénurie de développeurs COBOL

📈 Salaires explosent

Années 2000-2010 : Déclin Perçu

📉 Perception :

- "Langage mort"

- Moins enseigné dans les écoles

- Nouveaux langages (Java, C#, Python)

📊 Réalité :

- Toujours 220+ milliards de lignes en prod

- Base installée massive

- Coût de migration prohibitif

2020+ : Renaissance

📈 Regain d'intérêt :

- COVID-19 expose la dépendance

- Systèmes de chômage en COBOL surchargés

- Pénurie critique de compétences

- Salaires en hausse

Modernisation :

☁️ Cloud (AWS, Azure)

🐬 Conteneurisation (Docker)

🚀 APIs modernes

🖥️ Interfaces web

2.3 Le Mythe du "Langage Mort"

❌ **Idées Fausses Courantes**

Mythe #1 : "COBOL est obsolète"

❌ Faux !

Réalité :

- 43% des systèmes bancaires utilisent COBOL

- 95% des distributeurs automatiques

- 80% des transactions en personne

- 220+ milliards de lignes de code actif

Comparaison :

Python : ~100 millions de lignes totales sur GitHub

COBOL : 220,000 millions de lignes en PRODUCTION

Mythe #2 : "Personne n'apprend COBOL"

❌ Faux !

Réalité :

- IBM offre des cours COBOL gratuits

- Universités rajoutent COBOL au curriculum

- Bootcamps COBOL en émergence

- GitHub : 1000+ nouveaux repos COBOL/an

Demande :

📈 +20% par an depuis 2018

Mythe #3 : "COBOL va disparaître bientôt"

❌ Faux !

Réalité :

- Coût de migration : $1-5 millions par application

- Risque énorme de bugs

- ROI négatif dans la plupart des cas

Exemple :

Bank of America : 700 millions de lignes COBOL

Migration complète ? $3+ milliards, 10+ ans

Mythe #4 : "COBOL est limité aux mainframes"

❌ Faux !

Réalité moderne :

✅ COBOL sur Linux

✅ COBOL sur Windows

✅ COBOL sur Cloud (AWS, Azure, GCP)

✅ COBOL dans Docker

✅ COBOL appelant des APIs REST

✅ Microservices COBOL

✅ **La Vérité sur COBOL**

Forces Uniques :

1. **Précision Décimale**

```
* COBOL gère parfaitement l'argent
01 MONTANT PIC 9(10)V99.
* Pas d'erreurs d'arrondi comme en JavaScript/Python
```

2. **Lisibilité**

```
ADD SALAIRE TO TOTAL-SALAIRES
IF AGE IS GREATER THAN 65
  COMPUTE PENSION = SALAIRE * 0.70
END-IF
```

→ Même un non-programmeur comprend !

3. **Performance sur Batch**

Traitement de 10 millions d'enregistrements :

COBOL : 15 minutes

Python : 2+ heures

4. **Stabilité**

Uptime typique d'un système COBOL :

99.999% (5 minutes de downtime/an)

2.4 L'Écosystème COBOL Moderne

🛠️ Technologies Associées

Mainframes IBM z/OS

Le foyer traditionnel de COBOL

Composants :

- z/OS : Système d'exploitation
- DB2 : Base de données
- CICS : Gestionnaire de transactions
- IMS : Système de gestion de données
- JCL : Contrôle des jobs

Compilateurs Modernes

1. Micro Focus COBOL (Leader du marché)
   - Visual COBOL
   - Enterprise Developer
   - Support Cloud

2. IBM COBOL
   - Enterprise COBOL z/OS
   - COBOL for Linux

3. GnuCOBOL (Open Source)
   - Gratuit et open source
   - Compatible COBOL-85/2002
   - Linux/Windows/Mac

Outils de Développement

IDEs Modernes :

✅ Visual Studio Code + extensions COBOL

✅ Eclipse + COBOL plugins

✅ Micro Focus Visual COBOL IDE

✅ IBM Developer for z/OS

Debugging :

✅ Debuggers interactifs

✅ Code coverage tools

✅ Analyseurs statiques

Modernisation

Tendances actuelles :

1. Cloud Migration
   - Lift & shift vers AWS/Azure
   - Refactoring progressif
   - Approche hybride

2. APIs
   - COBOL exposé via REST APIs
   - Microservices COBOL
   - Intégration avec apps modernes

3. DevOps
   - CI/CD pour COBOL
   - Tests automatisés
   - Version control (Git)

4. Conteneurisation
   - COBOL dans Docker
   - Kubernetes orchestration

2.5 Carrières et Opportunités

💼 Profils de Postes

1. Développeur COBOL Mainframe

Responsabilités :

- Maintenance de programmes existants
- Développement de nouvelles fonctionnalités
- Debugging et support production
- Documentation

Salaire moyen : $75,000 - $95,000

Expérience : 0-5 ans

2. COBOL Modernization Specialist

Responsabilités :

- Migration vers le cloud
- Refactoring de code legacy
- Création d'APIs
- Architecture hybride

Salaire moyen : $95,000 - $120,000

Expérience : 5-10 ans

3. Architect COBOL/Mainframe

Responsabilités :

- Design système
- Stratégie de modernisation
- Mentorat d'équipe
- Décisions techniques

Salaire moyen : $120,000 - $180,000

Expérience : 10+ ans

4. COBOL Consultant

Responsabilités :

- Projets courts missions

- Expertise pointue

- Formation d'équipes

- Audit de code

Taux horaire : $100 - $250/h

Expérience : 5+ ans

🎓 **Parcours de Formation**

Débutant → Junior (6-12 mois)

Compétences :

✅ Syntaxe COBOL de base

✅ Fichiers séquentiels

✅ Structures de contrôle

✅ Debugging basique

Formation :

- Ce cours complet

- Projets personnels (5-10)

- Contributions open source

Junior → Intermédiaire (1-3 ans)

Compétences :

✅ VSAM (fichiers indexed)

✅ DB2 / SQL embarqué

✅ JCL avancé

✅ CICS basics

✅ Performance tuning

Formation :

- Cours spécialisés

- Projets professionnels

- Certification IBM (optionnel)

Intermédiaire → Senior (3-7 ans)

Compétences :

✅ Architecture système

✅ CICS avancé

✅ IMS

✅ Optimisation avancée

✅ Mentorat

Formation :

- Expérience terrain

- Projets complexes

- Veille technologique

Senior → Expert (7+ ans)

Compétences :

✅ Modernisation stratégique

✅ Cloud migration

✅ APIs et microservices

✅ Leadership technique

✅ Architecture d'entreprise

Formation :

- Conférences (SHARE, GSE)

- Certifications avancées

- Publications / speaking

---

3. Installation et Configuration de l'Environnement

3.1 Choix du Compilateur

🆓 **GnuCOBOL (Recommandé pour Débuter)**

Pourquoi GnuCOBOL ?

✅ 100% GRATUIT et open source

✅ Compatible Windows/Mac/Linux

✅ Conforme aux standards COBOL-85 et 2002

✅ Grande communauté

✅ Parfait pour apprendre

✅ Produit du code C (très performant)

Limitations :

❌ Pas de support CICS natif

❌ Pas d'intégration mainframe directe

❌ Moins d'outils enterprise

Verdict : 🌟 🌟 🌟 🌟 🌟 Excellent pour apprendre !

💰 **Alternatives Commerciales**

Micro Focus Visual COBOL

Prix : $3,000 - $10,000+/an

Avantages :

✅ IDE moderne intégré

✅ Support Cloud

✅ Debugging avancé

✅ Intégration .NET/Java

Pour : Entreprises et professionnels

IBM Enterprise COBOL

Prix : Variable (licence mainframe)

Avantages :

✅ Optimisé pour z/OS

✅ Support complet DB2/CICS

✅ Performance maximale

Pour : Développement mainframe professionnel

3.2 Installation sur Windows

📥 **Étape 1 : Télécharger GnuCOBOL**

Option A : Téléchargement Direct

1. Aller sur : https://sourceforge.net/projects/gnucobol/

2. Télécharger : gnucobol-3.x-win-installer.exe

3. Version recommandée : Dernière stable (3.2+)

Option B : Via Package Manager (Chocolatey)

```
# Installer Chocolatey d'abord

Set-ExecutionPolicy Bypass -Scope Process -Force

[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072

iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1')
)


# Installer GnuCOBOL

choco install gnucobol
```

🔧 Étape 2 : Installation

1. Double-cliquer sur l'installeur

2. Suivre l'assistant d'installation

3. Choisir le répertoire : C:\GnuCOBOL

4. Cocher "Add to PATH"

5. Installer les composants par défaut

6. Terminer l'installation

✅ Étape 3 : Vérification

Ouvrir Command Prompt (cmd)

Taper :

cobc --version

Sortie attendue :

cobc (GnuCOBOL) 3.2.0

Copyright (C) 2023 Free Software Foundation, Inc.

...

Si erreur "command not found" :

Ajouter manuellement au PATH :

1. Windows + Pause

2. Paramètres système avancés

3. Variables d'environnement

4. PATH → Modifier

5. Ajouter : C:\GnuCOBOL\bin

6. OK → Redémarrer le terminal

📝 Étape 4 : Éditeur de Texte

Option 1 : Visual Studio Code (Recommandé)

1. Télécharger : https://code.visualstudio.com/

2. Installer normalement

3. Ouvrir VS Code

4. Extensions (Ctrl+Shift+X)

5. Chercher "COBOL"

6. Installer : "COBOL Language Support" par BitLang

Configuration VS Code pour COBOL :

// Fichier : settings.json

```
{
  "cobol.margin": 72,
  "cobol.format": "fixed",
  "editor.rulers": [7, 11, 72],
  "editor.renderWhitespace": "all",
  "files.encoding": "utf8"
}
```

Option 2 : Notepad++

1. Télécharger : https://notepad-plus-plus.org/

2. Installer

3. Langage → COBOL (support natif)

3.3 Installation sur Mac

🍎 **Étape 1 : Installer Homebrew**

```
# Ouvrir Terminal
# Installer Homebrew si pas déjà fait
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

📥 **Étape 2 : Installer GnuCOBOL**

```
# Via Homebrew
brew install gnucobol


# Attendre l'installation...
```

Durée : 5-10 minutes

✅ **Étape 3 : Vérification**

Vérifier l'installation

cobc --version

Sortie attendue :

cobc (GnuCOBOL) 3.2.0

...

📝 **Étape 4 : Éditeur**

Visual Studio Code :

Installer VS Code

brew install --cask visual-studio-code

Ouvrir VS Code

code .

Installer l'extension COBOL (même procédure que Windows)

3.4 Installation sur Linux

🐧 **Ubuntu/Debian**

Mettre à jour les paquets

sudo apt update

Installer les dépendances

sudo apt install build-essential

Installer GnuCOBOL

sudo apt install gnucobol

```
# Vérifier
cobc --version
```

🎩 **Fedora/RHEL/CentOS**

```
# Installer les dépendances
sudo dnf install gcc make

# Installer GnuCOBOL
sudo dnf install gnucobol

# Vérifier
cobc --version
```

📦 **Compilation depuis les sources (Toutes distros)**

```
# Télécharger les sources
cd /tmp
wget https://ftp.gnu.org/gnu/gnucobol/gnucobol-3.2.tar.gz

# Extraire
tar -xzf gnucobol-3.2.tar.gz
cd gnucobol-3.2

# Compiler
./configure
make
sudo make install

# Vérifier
cobc --version
```

3.5 Premier Programme : "Hello World"

📝 Créer le Fichier

Windows :

```
# Créer un dossier pour vos programmes
mkdir C:\COBOL
cd C:\COBOL

# Créer le fichier avec VS Code
code HELLO.cob
```

Mac/Linux :

```
# Créer un dossier
mkdir ~/cobol
cd ~/cobol

# Créer le fichier
code HELLO.cob
```

💻 Code du Programme

```
      ******************************************************************
      * Programme : HELLO.cob
      * Description : Premier programme - Affiche "Hello World"
      * Auteur : Ton Nom
      * Date : 02/11/2024
      ******************************************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       AUTHOR. TON-NOM.
       DATE-WRITTEN. 02/11/2024.

       ENVIRONMENT DIVISION.
```

```
    DATA DIVISION.


    PROCEDURE DIVISION.
      DISPLAY "Hello World!".
      DISPLAY "Bienvenue dans le monde du COBOL!".
      STOP RUN.
```

⚠️ **IMPORTANT : Respect des Colonnes COBOL**

Colonnes 1-6 : Numéros de séquence (optionnel, souvent vide)

Colonne 7 : Indicateur (*, -, /) ou espace

Colonnes 8-11 : Area A (pour les divisions, sections, paragraphes)

Colonnes 12-72 : Area B (pour les instructions)

Colonnes 73-80 : Identification (optionnel, ignoré)

Exemple visuel :

```
12345678901234567890123456789O... (numéros de colonnes)
    *Commentaire (col 7 = *)
     PROGRAM-ID. HELLO. (Area A commence col 8)
       DISPLAY "Test". (Area B commence col 12)
```

🔨 **Compiler le Programme**

Windows (cmd) :

```
cd C:\COBOL
cobc -x -free HELLO.cob
```

Mac/Linux (terminal) :

```
cd ~/cobol
cobc -x -free HELLO.cob
```

Explication des options :

-x : Crée un exécutable

-free : Format libre (ignore colonnes strictes)

-fixed : Format fixe (respecte colonnes) [par défaut]

Sortie attendue :

(Aucun message = succès !)

Fichier créé : HELLO (ou HELLO.exe sur Windows)

▶️ Exécuter le Programme

Windows :

HELLO.exe

Ou simplement :

HELLO

Mac/Linux :

./HELLO

Résultat attendu :

Hello World!

Bienvenue dans le monde du COBOL!

🎉 Félicitations !

Tu viens de :

1. ✅ Installer un compilateur COBOL
2. ✅ Écrire ton premier programme
3. ✅ Compiler avec succès
4. ✅ Exécuter le programme

🚀 Tu es maintenant prêt pour apprendre COBOL sérieusement !

3.6 Configuration IDE Avancée (Optionnel)

🎨 Extensions VS Code Recommandées

1. COBOL Language Support

ID : bitlang.cobol

Fonctionnalités :

- Coloration syntaxique

- Snippets

- Vérification des colonnes

2. COBOL Copybook Formatter

ID : rechinformatica.rech-cobol-editor

Fonctionnalités :

- Formatting automatique

- Outline des programmes

- Support copybooks

3. Remote Development (pour mainframe)

ID : ms-vscode-remote.remote-ssh

Fonctionnalités :

- Connexion SSH à mainframe

- Édition directe

⚙️ **Configuration tasks.json**

Créer .vscode/tasks.json :

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Compile COBOL",
      "type": "shell",
      "command": "cobc",
      "args": [
        "-x",
        "-free",
        "${file}"
      ],
```

```
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "presentation": {
        "echo": true,
        "reveal": "always",
        "focus": false,
        "panel": "shared"
      }
    },
    {
      "label": "Run COBOL",
      "type": "shell",
      "command": "${fileDirname}/${fileBasenameNoExtension}",
      "dependsOn": "Compile COBOL",
      "group": {
        "kind": "test",
        "isDefault": true
      }
    }
  ]
}
```

Utilisation :

Ctrl+Shift+B : Compiler

Ctrl+Shift+T : Compiler et exécuter

🐛 **Configuration launch.json (Debugging)**

Créer .vscode/launch.json :

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug COBOL",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${fileDirname}",
      "environment": [],
      "externalConsole": true
    }
  ]
}
```

3.7 Dépannage Courant

❌ **Problème : "cobc: command not found"**

Solution Windows :

```
# Vérifier l'installation
dir "C:\GnuCOBOL\bin"

# Ajouter au PATH
setx PATH "%PATH%;C:\GnuCOBOL\bin"

# Redémarrer le terminal
```

Solution Mac/Linux :

```
# Trouver où est installé cobc
```

which cobc

Si pas trouvé, réinstaller

brew reinstall gnucobol # Mac

sudo apt install --reinstall gnucobol # Linux

❌ **Problème : Erreurs de Compilation**

Erreur : "Invalid character in column..."

Cause : Colonnes COBOL non respectées

Solution :

1. Utiliser l'option -free

2. OU respecter les colonnes (Area A/B)

3. Vérifier l'encodage (UTF-8)

Erreur : "PROGRAM-ID not found"

Cause : Structure du programme incorrecte

Solution :

Vérifier l'ordre des divisions :

1. IDENTIFICATION DIVISION

2. ENVIRONMENT DIVISION (optionnel)

3. DATA DIVISION (optionnel)

4. PROCEDURE DIVISION

❌ **Problème : Programme se Ferme Immédiatement**

Windows :

* Ajouter ACCEPT avant STOP RUN

```
PROCEDURE DIVISION.
    DISPLAY "Hello World!".
    DISPLAY "Appuyez sur Enter pour quitter".
```

```
ACCEPT WS-DUMMY.  *> Attend une entrée
STOP RUN.
```

Ou exécuter depuis le terminal :

```
cmd /k HELLO.exe
```

---

4. Structure d'un Programme COBOL

4.1 Vue d'Ensemble

Un programme COBOL est comme une **maison** avec différentes pièces :

🏠 PROGRAMME COBOL

├── 🚪 IDENTIFICATION DIVISION (Carte d'identité)

├── 🌍 ENVIRONMENT DIVISION (Où habite le programme)

├── 📦 DATA DIVISION (Affaires et meubles)

└── 🎬 PROCEDURE DIVISION (Activités quotidiennes)

📐 **Structure Obligatoire**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. NOM-PROGRAMME.

[ENVIRONMENT DIVISION.]
[Configuration et fichiers]

[DATA DIVISION.]
[Variables et structures de données]

PROCEDURE DIVISION.
[Instructions du programme]
STOP RUN.
```

Points Clés :

✅ IDENTIFICATION DIVISION : OBLIGATOIRE

✅ PROCEDURE DIVISION : OBLIGATOIRE

⚠️ ENVIRONMENT DIVISION : Optionnel (sauf si fichiers)

⚠️ DATA DIVISION : Optionnel (sauf si variables)

4.2 Les 4 Divisions en Détail

🚪 **1. IDENTIFICATION DIVISION**

Rôle : Identifier le programme (nom, auteur, date, etc.)

Clauses Principales :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CALC-SALAIRE.
AUTHOR. JEAN DUPONT.
DATE-WRITTEN. 02/11/2024.
DATE-COMPILED.
INSTALLATION. BANQUE XYZ.
SECURITY. CONFIDENTIEL.
REMARKS. Programme de calcul de salaire mensuel.
```

Explication Clause par Clause :

Clause	Obligatoire ?	Description
PROGRAM-ID	✅ OUI	Nom du programme (max 30 caractères)
AUTHOR	❌ Non	Nom de l'auteur
DATE-WRITTEN	❌ Non	Date d'écriture initiale
DATE-COMPILED	❌ Non	Date de compilation (auto)
INSTALLATION	❌ Non	Où le programme tourne
SECURITY	❌ Non	Niveau de sécurité
REMARKS	❌ Non	Commentaires généraux

Règles pour PROGRAM-ID :

✅ Bon : CALC-SALAIRE, PROG01, UPDATE-CLIENT

❌ Mauvais :

- CALCUL SALAIRE (espace non autorisé)

- PROG#01 (caractères spéciaux sauf -)

- calcul-salaire (doit être en MAJUSCULES)

Exemple Complet :

```
      ******************************************************************
      * Programme de Gestion de Compte Bancaire
      ******************************************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID. GESTION-COMPTE.
       AUTHOR. PIERRE MARTIN.
       DATE-WRITTEN. 02/11/2024.
       DATE-COMPILED.
       INSTALLATION. BANQUE NATIONALE.
       SECURITY. NIVEAU-3.
       REMARKS.
           Ce programme gère les opérations de base sur les comptes :
           - Consultation de solde
           - Dépôt
           - Retrait
           - Historique des transactions
```

🌎 2. ENVIRONMENT DIVISION

Rôle : Décrire l'environnement d'exécution (ordinateur, fichiers)

Structure :

```
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       [Configuration de l'ordinateur]
```

INPUT-OUTPUT SECTION.

[Déclaration des fichiers]

Exemple Simple (sans fichiers) :

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
```

Exemple avec Fichiers :

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT FICHIER-CLIENTS
    ASSIGN TO "CLIENTS.DAT"
    ORGANIZATION IS SEQUENTIAL
    ACCESS MODE IS SEQUENTIAL
    FILE STATUS IS WS-FILE-STATUS.

  SELECT FICHIER-RAPPORT
    ASSIGN TO "RAPPORT.TXT"
    ORGANIZATION IS LINE SEQUENTIAL.
```

Explication :

```
SELECT nom-fichier-logique     *> Nom utilisé dans le programme
  ASSIGN TO "nom-physique"    *> Nom réel du fichier
```

```
ORGANIZATION IS type      *> SEQUENTIAL, INDEXED, RELATIVE
ACCESS MODE IS mode       *> SEQUENTIAL, RANDOM, DYNAMIC
FILE STATUS IS variable   *> Variable pour les erreurs (00=ok)
```

Types d'Organisation :

Type	Usage	Exemple
SEQUENTIAL	Fichiers lus séquentiellement	Logs, rapports
INDEXED	Accès direct par clé	Fichier clients (par ID)
RELATIVE	Accès par numéro d'enregistrement	Moins utilisé
LINE SEQUENTIAL	Fichiers texte (une ligne = un enregistrement)	Fichiers CSV

📦 3. DATA DIVISION

Rôle : Déclarer TOUTES les données utilisées par le programme

Structure :

```
DATA DIVISION.
FILE SECTION.
[Structure des fichiers]

WORKING-STORAGE SECTION.
[Variables de travail]

LOCAL-STORAGE SECTION.
[Variables locales (si sous-programmes)]

LINKAGE SECTION.
[Paramètres passés entre programmes]
```

Sections Principales :

FILE SECTION : Structure des fichiers

```
FILE SECTION.
```

```
FD  FICHIER-CLIENTS.
01  ENREG-CLIENT.
   05 CLI-ID         PIC 9(6).
   05 CLI-NOM        PIC X(30).
   05 CLI-SOLDE      PIC 9(7)V99.
```

WORKING-STORAGE SECTION : Variables de travail

```
WORKING-STORAGE SECTION.
*Variables simples
01  WS-COMPTEUR       PIC 9(5) VALUE 0.
01  WS-TOTAL          PIC 9(9)V99 VALUE ZERO.
01  WS-MESSAGE        PIC X(50).

*Variables structurées
01  WS-DATE-DU-JOUR.
   05 WS-ANNEE       PIC 9(4).
   05 WS-MOIS        PIC 99.
   05 WS-JOUR        PIC 99.
```

💡 **Différence importante :**

WORKING-STORAGE :

✅ Persiste entre appels (si sous-programme)

✅ Peut avoir VALUE

LOCAL-STORAGE :

❌ Réinitialisé à chaque appel

✅ Plus efficace en mémoire

🎬 **4. PROCEDURE DIVISION**

Rôle : Contient la logique du programme (le code exécutable)

Structure Typique :

```
       PROCEDURE DIVISION.
       MAIN-PROCEDURE.
      ******************************************************************
      * Programme principal
      ******************************************************************
           PERFORM INITIALISATION.
           PERFORM TRAITEMENT.
           PERFORM TERMINAISON.
           STOP RUN.

       INITIALISATION.
      ******************************************************************
      * Initialiser les variables et ouvrir les fichiers
      ******************************************************************
           DISPLAY "Début du programme".
           MOVE 0 TO WS-COMPTEUR.
           MOVE ZERO TO WS-TOTAL.

       TRAITEMENT.
      ******************************************************************
      * Logique principale
      ******************************************************************
           DISPLAY "Traitement en cours...".
           ADD 1 TO WS-COMPTEUR.

       TERMINAISON.
      ******************************************************************
      * Fermer les fichiers et afficher le résumé
```

```
      ************************************************************

          DISPLAY "Fin du programme".

          DISPLAY "Nombre d'opérations : " WS-COMPTEUR.
```

Organisation en Paragraphes :

```
PROCEDURE DIVISION.

   Paragraphe-Principal.     *> Point d'entrée

      PERFORM Sous-Procedure.

      STOP RUN.


   Sous-Procedure.          *> Comme une "fonction"

      DISPLAY "Je suis une sous-procedure".
```

Conventions de Nommage :

✅ Bon style :

- 000-MAIN-PROCEDURE
- 100-INITIALISATION
- 200-TRAITEMENT-PRINCIPAL
- 300-TRAITEMENT-FICHIERS
- 900-TERMINAISON

💡 Les numéros aident à organiser et voir la structure !

4.3 Programme Complet Commenté

```
      ****************************************************************

      * PROGRAMME : CALC-MOYENNE

      * DESCRIPTION : Calcule la moyenne de 3 notes

      * AUTEUR : Ton Nom

      * DATE : 02/11/2024

      ****************************************************************
```

```
*=================================================================*
* IDENTIFICATION : QUI EST CE PROGRAMME ?
*=================================================================*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CALC-MOYENNE.
 AUTHOR. TON-NOM.
 DATE-WRITTEN. 02/11/2024.

*=================================================================*
* ENVIRONMENT : OÙ ET COMMENT IL TOURNE ?
*=================================================================*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-370.
 OBJECT-COMPUTER. IBM-370.

*=================================================================*
* DATA : QUELLES DONNÉES IL UTILISE ?
*=================================================================*
 DATA DIVISION.
 WORKING-STORAGE SECTION.

*--- Variables pour les notes ---
 01  WS-NOTE1        PIC 99V99.
 01  WS-NOTE2        PIC 99V99.
 01  WS-NOTE3        PIC 99V99.

*--- Calculs ---
```

```
 01  WS-TOTAL          PIC 999V99.
 01  WS-MOYENNE        PIC 99V99.

*--- Affichage ---
 01  WS-MESSAGE        PIC X(50).
 01  WS-RESULTAT       PIC Z9.99.  *> Format d'affichage

*================================================================*
* PROCEDURE : QUE FAIT-IL ? (LE CODE EXÉCUTABLE)
*================================================================*
 PROCEDURE DIVISION.

*----------------------------------------------------------------
* Programme principal
*----------------------------------------------------------------
 000-MAIN-PROCEDURE.
    DISPLAY "======================================".
    DISPLAY "    CALCUL DE MOYENNE DE 3 NOTES    ".
    DISPLAY "======================================".
    DISPLAY " ".

    PERFORM 100-SAISIE-NOTES.
    PERFORM 200-CALCUL-MOYENNE.
    PERFORM 300-AFFICHAGE-RESULTAT.

    DISPLAY " ".
    DISPLAY "Programme terminé avec succès.".
    STOP RUN.
```

```
*-----------------------------------------------------------------
* Saisie des 3 notes
*-----------------------------------------------------------------
 100-SAISIE-NOTES.
    DISPLAY "Entrez la note 1 (sur 20) : " WITH NO ADVANCING.
    ACCEPT WS-NOTE1.

    DISPLAY "Entrez la note 2 (sur 20) : " WITH NO ADVANCING.
    ACCEPT WS-NOTE2.

    DISPLAY "Entrez la note 3 (sur 20) : " WITH NO ADVANCING.
    ACCEPT WS-NOTE3.

    DISPLAY " ".

*-----------------------------------------------------------------
* Calcul de la moyenne
*-----------------------------------------------------------------
 200-CALCUL-MOYENNE.
    COMPUTE WS-TOTAL = WS-NOTE1 + WS-NOTE2 + WS-NOTE3.
    COMPUTE WS-MOYENNE = WS-TOTAL / 3.

*-----------------------------------------------------------------
* Affichage du résultat
*-----------------------------------------------------------------
 300-AFFICHAGE-RESULTAT.
    MOVE WS-MOYENNE TO WS-RESULTAT.
```

```
       DISPLAY "========================================".
       DISPLAY " ".
       DISPLAY "Note 1      : " WS-NOTE1.
       DISPLAY "Note 2      : " WS-NOTE2.
       DISPLAY "Note 3      : " WS-NOTE3.
       DISPLAY "-------------------------".
       DISPLAY "Total       : " WS-TOTAL.
       DISPLAY "MOYENNE     : " WS-RESULTAT.
       DISPLAY " ".

       IF WS-MOYENNE >= 10
          DISPLAY "Résultat    : ADMIS ✓"
       ELSE
          DISPLAY "Résultat    : RECALÉ ✗"
       END-IF.

       DISPLAY "========================================".
```

📝 **Pour Exécuter :**

```
# Compiler
cobc -x -free CALC-MOYENNE.cob

# Exécuter
./CALC-MOYENNE

# Exemple d'exécution :
========================================
   CALCUL DE MOYENNE DE 3 NOTES
```

```
========================================

Entrez la note 1 (sur 20) : 15.50
Entrez la note 2 (sur 20) : 12.00
Entrez la note 3 (sur 20) : 16.75

========================================

Note 1      : 15.50
Note 2      : 12.00
Note 3      : 16.75
-------------------------
Total       : 44.25
MOYENNE     : 14.75

Résultat    : ADMIS ✓
========================================

Programme terminé avec succès.
```

---

5. Les Divisions COBOL Expliquées

5.1 Pourquoi 4 Divisions ?

Philosophie de COBOL :

"Séparer les préoccupations pour plus de clarté"

📌 IDENTIFICATION : "Qui je suis"

🌍 ENVIRONMENT : "Où je vis"

📦 DATA   : "Ce que j'ai"

🎬 PROCEDURE : "Ce que je fais"

Comparaison avec d'autres langages :

```
# Python (tout mélangé)
def calcul_moyenne():
    # Configuration et données ensemble
    note1 = float(input("Note 1: "))
    note2 = float(input("Note 2: "))
    moyenne = (note1 + note2) / 2
    print(f"Moyenne: {moyenne}")
*> COBOL (bien séparé)
IDENTIFICATION DIVISION.  → Métadonnées
ENVIRONMENT DIVISION.     → Config
DATA DIVISION.            → Variables
PROCEDURE DIVISION.       → Logique
```

Avantages de cette structure :

✅ Lisibilité : Facile de trouver où est déclaré quoi

✅ Maintenance : Modifications localisées

✅ Documentation : Structure auto-documentée

✅ Standards : Toujours la même organisation

5.2 IDENTIFICATION DIVISION en Profondeur

📋 **Toutes les Clauses Disponibles**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MON-PROGRAMME
    IS INITIAL PROGRAM     *> Programme réinitialisé à chaque appel
    IS COMMON PROGRAM      *> Partagé entre modules
    IS RECURSIVE.          *> Peut s'appeler lui-même

AUTHOR. JEAN DUPONT - EQUIPE DEV.
```

```
       DATE-WRITTEN. 02/11/2024.
       DATE-COMPILED.  *> Rempli automatiquement lors de la compilation

       INSTALLATION.
           BANQUE XYZ - DÉPARTEMENT IT
           SERVEUR PROD-01.

       SECURITY.
           NIVEAU CONFIDENTIEL
           ACCES RESTREINT AU PERSONNEL AUTORISE.

       REMARKS.
           Programme de calcul de prêt bancaire.
           Utilise la formule standard d'amortissement.
           Version 2.1 - Ajout gestion des assurances.
```

💡 **Clauses Spéciales PROGRAM-ID**

IS INITIAL :

```
PROGRAM-ID. SOUS-PROG IS INITIAL.

*> Effet : Le programme est réinitialisé à chaque appel
*> Usage : Sous-programmes qui ne doivent PAS garder d'état
```

IS RECURSIVE :

```
PROGRAM-ID. FACTORIELLE IS RECURSIVE.

*> Effet : Le programme peut s'appeler lui-même
*> Usage : Algorithmes récursifs (factorielle, Fibonacci)
```

Exemple Recursion :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. FACTORIELLE IS RECURSIVE.

DATA DIVISION.
WORKING-STORAGE SECTION.
01  WS-NOMBRE       PIC 9(2).
01  WS-RESULTAT     PIC 9(18).

LINKAGE SECTION.
01  LS-N            PIC 9(2).
01  LS-FACT         PIC 9(18).

PROCEDURE DIVISION USING LS-N RETURNING LS-FACT.
   IF LS-N <= 1
      MOVE 1 TO LS-FACT
   ELSE
      SUBTRACT 1 FROM LS-N GIVING WS-NOMBRE
      CALL "FACTORIELLE" USING WS-NOMBRE
         RETURNING WS-RESULTAT
      MULTIPLY LS-N BY WS-RESULTAT GIVING LS-FACT
   END-IF.
   GOBACK.
```

📝 **Bonnes Pratiques**

✅ À FAIRE :

- Nom de programme descriptif (CALC-SALAIRE, pas PROG01)
- Auteur avec email ou équipe
- Date au format YYYY-MM-DD ou JJ/MM/AAAA
- REMARKS pour expliquer l'objectif global

❌ À ÉVITER :

- Noms génériques (PROG, TEST, TEMP)

- Informations obsolètes (ne pas mettre à jour)

- Remarques trop techniques (mettre dans les commentaires)

5.3 ENVIRONMENT DIVISION en Profondeur

🔧 **CONFIGURATION SECTION Détaillée**

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

SOURCE-COMPUTER. IBM-370
   WITH DEBUGGING MODE.    *> Active le mode debug

OBJECT-COMPUTER. IBM-370
   PROGRAM COLLATING SEQUENCE IS EBCDIC
   MEMORY SIZE IS 2048 WORDS
   SEGMENT-LIMIT IS 64.

SPECIAL-NAMES.
   DECIMAL-POINT IS COMMA   *> Utilise , au lieu de .
   CURRENCY SIGN IS "€"     *> Symbole monétaire
   CONSOLE IS CRT
   C01 IS NEW-PAGE         *> Saut de page à l'impression
   C02 IS NEW-LINE.        *> Nouvelle ligne
```

Explication Clauses :

Clause	Usage	Exemple
WITH DEBUGGING MODE	Active les lignes de debug	D DISPLAY "Debug: " WS-VAR
PROGRAM COLLATING SEQUENCE	Ordre de tri (ASCII/EBCDIC)	Important pour les comparaisons
DECIMAL-POINT IS COMMA	Notation européenne	12,50 au lieu de 12.50
CURRENCY SIGN	Symbole personnalisé	€123,45 au lieu de $123.45

📁 INPUT-OUTPUT SECTION Détaillée

Fichier Sequential :

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT FICHIER-INPUT
    ASSIGN TO "C:\DATA\INPUT.DAT"  *> Chemin Windows
    ORGANIZATION IS SEQUENTIAL
    ACCESS MODE IS SEQUENTIAL
    FILE STATUS IS WS-INPUT-STATUS.
```

Fichier Indexed (VSAM) :

```
    SELECT FICHIER-CLIENTS
      ASSIGN TO "CLIENTS"          *> Nom logique (MVS)
      ORGANIZATION IS INDEXED
      ACCESS MODE IS DYNAMIC       *> SEQUENTIAL, RANDOM, DYNAMIC
      RECORD KEY IS CLI-ID         *> Clé primaire
      ALTERNATE RECORD KEY IS CLI-NOM *> Clé secondaire
        WITH DUPLICATES
      FILE STATUS IS WS-CLI-STATUS.
```

Fichier Relative :

```
    SELECT FICHIER-STOCK
      ASSIGN TO "STOCK.DAT"
```

ORGANIZATION IS RELATIVE

ACCESS MODE IS RANDOM

RELATIVE KEY IS WS-RECORD-NUM

FILE STATUS IS WS-STOCK-STATUS.

📊 **FILE STATUS Codes**

*> File Status à 2 chiffres : XY

*> X = Catégorie, Y = Détail

00 : Succès

02 : Duplicate key (pour alternate key avec duplicates)

10 : Fin de fichier (EOF)

21 : Erreur de séquence

22 : Duplicate key (non autorisé)

23 : Enregistrement non trouvé

24 : Espace disque insuffisant

30 : Erreur permanente

34 : Erreur d'espace (fichier plein)

35 : Fichier inexistant

37 : Mode d'accès incorrect

39 : Conflit d'attributs

90-99 : Erreurs spécifiques au système

Exemple d'utilisation :

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  WS-FILE-STATUS       PIC XX.
   88 WS-FILE-OK        VALUE "00".
   88 WS-FILE-EOF       VALUE "10".
   88 WS-FILE-NOT-FOUND  VALUE "35".
```

```
PROCEDURE DIVISION.
    OPEN INPUT FICHIER-CLIENTS.

    IF NOT WS-FILE-OK
        DISPLAY "Erreur ouverture: " WS-FILE-STATUS
        STOP RUN
    END-IF.

    PERFORM UNTIL WS-FILE-EOF
        READ FICHIER-CLIENTS
            AT END
                SET WS-FILE-EOF TO TRUE
            NOT AT END
                PERFORM TRAITER-ENREGISTREMENT
        END-READ
    END-PERFORM.
```

(La suite du cours continue dans la partie 2 avec les chapitres 6-10...)

🎯 **Points Couverts jusqu'ici :**

- ✅ Introduction complète au COBOL
- ✅ Histoire et contexte détaillés
- ✅ Installation pas à pas (Windows/Mac/Linux)
- ✅ Structure des 4 divisions expliquée
- ✅ Premier programme fonctionnel
- ✅ Exemples pratiques et commentés

📚 **À venir dans la Partie 2 :**

- Types de données et variables
- PICTURE clauses en détail
- Opérations arithmétiques
- Structures conditionnelles
- Boucles et itérations
- Et bien plus encore !

Durée d'étude de cette Partie 1 : 6-8 heures

Continue vers la Partie 2 quand tu es prêt ! 🚀