C32 Développer à l'aide d'un langage de programmation procédural

Le candidat est invité à rechercher une illustration, tirée de son activité, de chaque aspect technique de la compétence correspondant présenté dans cette fiche. Cette illustration pourra être mentionnée dans le dossier, et sera présentée lors de l'épreuve.

La présente fiche se présente sous la forme d'une énumération, elle n'est ni un format ni un plan pour l'épreuve ou le dossier d'activité.

Les termes en gras appartiennent au vocabulaire technique de référence de la compétence, ils doivent donc être naturellement employés par le candidat lors de l'épreuve

Le langage procédural :
Permet de **structurer** un programme en le découpant en **fonctions** et **procédures** et de **réutiliser** une **séquence d'instruction** définie et **testée**.
Facilite le développement des applications.

1. Les sous programmes :

Les **procédures** peuvent **renvoyer aucune ou plusieurs valeurs**.

Les **fonctions** sont des sous-programmes **typés** qui renvoient toujours **au moins une valeur**.

Les **paramètres formels** sont définis **dans l'en-tête** du sous-programme entre parenthèses après le nom du sous-programme et avec, pour chaque paramètre son nom et son type.

Utilisables dans les instructions du sous-programme sans être **déclarés** et sans être **initialisés**.

Les valeurs des paramètres **effectifs** sont précisées **au moment de l'appel**.
Soit des noms de **variables** de **l'appelant**, soit des littéraux.

Le **nombre**, l'**ordre** et les **types** des paramètres doivent être **identiques** dans l'en-tête du sous-programme et dans l'appelant.

Signature : Le nombre et le type des **arguments** d'une procédure ou fonction (dans ce cas, la signature précisera le type de la valeur retournée par la fonction).

Le passage de paramètres : La manière dont les paramètres sont passés est définie dans l'en-tête du sous-programme.

Les paramètres passés **par valeur** (ByVal en VB) : paramètres **données** (D : en algo).
Toute modification de cette valeur **dans le sous-programme** est sans effet sur la valeur du paramètre dans le programme **appelant**.

Exemple à fournir par le candidat

Programme appelant	Sous-programme

Les paramètres passés **par référence** (ByRef en VB) : **données/résultats** (D/R : valeur au moment de l'appel utilisée dans le sous programme) ou **résultats** (R : valeur déterminée par le sous-programme).
Toute modification de la valeur du paramètre **dans le sous-programme** est répercutée sur la valeur du paramètre dans le programme **appelant**.

Exemple à fournir par le candidat

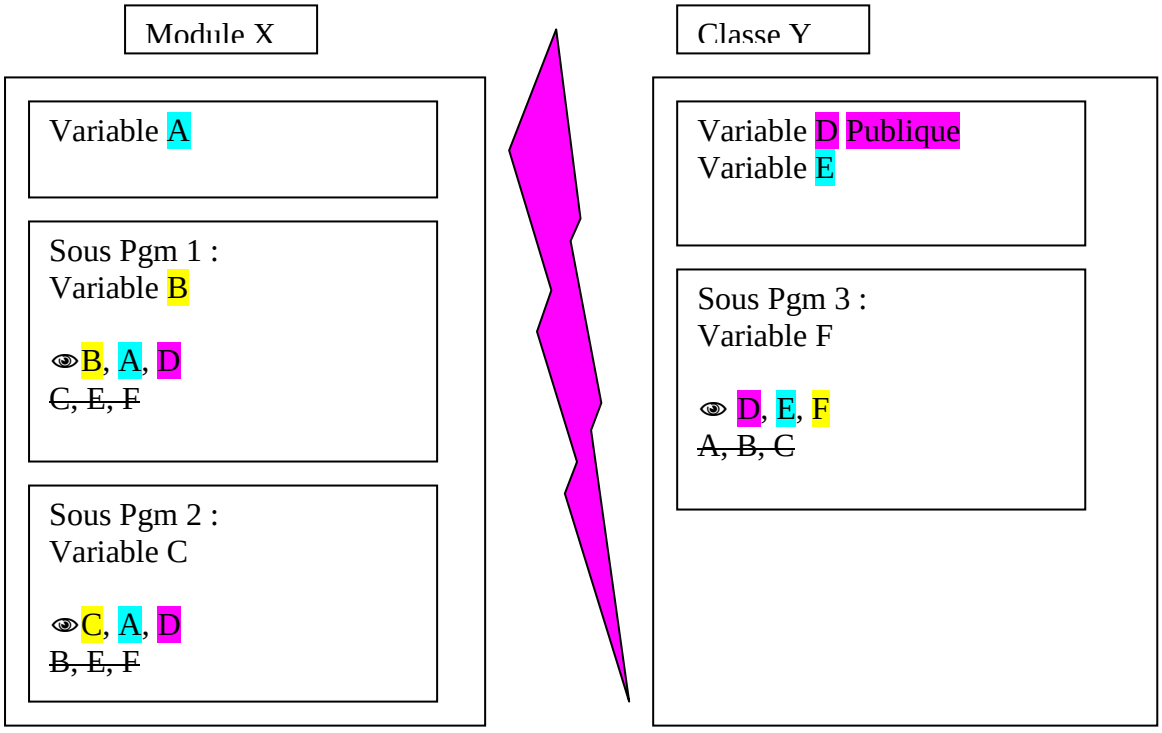
Programme appelant	Sous-programme

2. Portée des variables

La portée d'une variable (plus généralement d'un objet) définit les parties du programme qui en connaissent l'existence.

Dans l'exemple ci-dessous :
- la variable A est globale au module X, la variable E est globale à la classe Y
- la variable B est locale au sous programme 1, la variable C est locale au sous programme 2, la variable F est locale au sous programme 3
- la variable D est globale à la classe Y, et publique : elle est donc aussi visible par tout module, classe, et donc sous programme de l'application

Remarque concernant les « bonnes pratiques en programmation »:
Pour sécuriser le code (moins de bugs « d'effet de bord ») et une meilleure lisibilité, on limite au maximum le recours aux variables globales.



3. Types structurés

Les **structures de données** peuvent être soit **des tableaux, des fichiers, des listes chaînées, des collections, des arbres binaires,** etc. Elles servent à stocker les données du programme afin de les **manipuler (parcours, recherche, tri,** etc.) efficacement dans le code.

4. Enrichissements possibles

- Recours au **débogueur**
- Recours à **l'aide (en ligne)**
- **Récursivité**
- Etc.