



Ressourcesinformatiques

Version numérique

OFFERTE !

www.editions-eni.fr

Apprendre à programmer avec **ABAP**

Les fondamentaux du développement sur SAP (avec exercices et corrigés)

Nicolas PONTIER

Fichiers complémentaires
à télécharger



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RIABAP** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1 Qu'est-ce que la programmation ?

- 1. Utilité et fonctionnement 9
- 2. Les étapes de la programmation 11
 - 2.1 La demande 11
 - 2.2 L'analyse 11
 - 2.3 L'algorithme 12
 - 2.4 Le développement 14
 - 2.5 Les tests unitaires 14
 - 2.6 La documentation technique 16

Chapitre 2 SAP

- 1. L'architecture SAP 17
 - 1.1 Qu'est-ce que SAP ? 17
 - 1.2 Architecture 21
- 2. Les modules 23

Chapitre 3

Premiers pas sur SAP

1. Détail d'un mandant SAP	27
2. Connexion et fenêtre d'accueil	31
3. Premier programme ABAP.....	44
3.1 Les transactions.....	44
3.2 'Hello World'	48
3.3 Gestion des ordres de transport	61
4. Exercice.....	68

Chapitre 4

Instructions basiques ABAP

1. Variables et constantes	69
2. Opérations arithmétiques	74
3. Opérations sur variable texte	78
3.1 CONCATENATE	78
3.2 CONDENSE	80
3.3 SPLIT	81
3.4 FIND	83
3.5 REPLACE	86
3.6 TRANSLATE.....	89
3.7 CLEAR	90
3.8 Traitement d'un string	91
4. Instructions conditionnelles	93
4.1 IF... ENDIF.....	93
4.1.1 Pour tout type de données.....	94
4.1.2 Données chaînes de caractères	97
4.2 CASE... ENDCASE	103
5. Les boucles	104
5.1 DO... ENDDO	104
5.2 WHILE... ENDWHILE	108
5.3 Boucle infinie.....	110

6. Les erreurs ABAP 112
 7. Exercice..... 118

Chapitre 5
Dictionnaire de données (DDIC)

1. Définition..... 119
 2. Les domaines 122
 2.1 Définition 122
 2.2 Création d'un domaine..... 129
 3. Éléments de données 131
 3.1 Définition 131
 3.2 Création d'un élément de données 136
 4. Aide à la recherche..... 138
 4.1 Définition 138
 4.2 Création d'une aide à la recherche 148
 5. Structures et Tables..... 152
 5.1 Les tables 152
 5.1.1 Barre d'outils 153
 5.1.2 Les onglets 160
 5.1.3 Gestion de la table 170
 5.1.4 Création d'une table 176
 5.2 DDIC supplémentaires..... 183
 5.2.1 Éléments de données 183
 5.2.2 Tables 185
 5.3 Les structures 197
 5.3.1 Définition..... 197
 5.3.2 Création d'un append 200
 5.4 Exercice : création d'une table 203

Chapitre 6

Les tables internes

1. Caractéristiques	205
1.1 Définition	205
1.2 Les attributs de la table interne	207
1.2.1 Les types de tables internes	207
1.2.2 La clé primaire	209
1.2.3 La clé secondaire	210
2. Travail sur les données	212
2.1 Ajout d'enregistrements	212
2.1.1 INSERT	212
2.1.2 APPEND	214
2.1.3 COLLECT	217
2.2 MODIFY	218
2.2.1 MODIFY avec index	218
2.2.2 MODIFY TABLE	220
2.2.3 MODIFY avec WHERE	221
2.3 DELETE	223
2.3.1 Suppression d'une seule ligne	223
2.3.2 Suppression de plusieurs lignes	226
2.3.3 Suppression de lignes en doublon	227
3. Organisation et Lecture	230
3.1 SORT	230
3.2 READ TABLE	232
3.3 LOOP	238
3.3.1 Caractéristiques principales	238
3.3.2 Paramètres AT	240
4. Outils	244
4.1 FIND IN TABLE	244
4.2 REPLACE IN TABLE	246
4.3 DESCRIBE TABLE	249
4.4 DIVERS	250
5. Exercice	251

Chapitre 7
Les requêtes SQL

- 1. Définition. 253
- 2. SELECT 255
 - 2.1 SELECT 256
 - 2.1.1 SINGLE [FOR UPDATE] 256
 - 2.1.2 DISTINCT 259
 - 2.1.3 Col AS alias 260
 - 2.1.4 Les agrégats 260
 - 2.1.5 Les expressions SQL. 264
 - 2.2 FROM 273
 - 2.3 INTO 280
 - 2.4 WHERE 287
 - 2.5 Autres options. 299
- 3. INSERT 304
- 4. UPDATE. 306
- 5. MODIFY 311
- 6. DELETE 311
- 7. Index. 313
- 8. Exercice. 317

Chapitre 8
Les fonctions

- 1. Type de fonction 319
- 2. Paramètres et appel de la fonction 331
- 3. Création d'une fonction 342
- 4. Exercice. 349

Chapitre 9

Les classes

1. Introduction à l'ABAP Objet	351
2. Caractéristiques des classes	352
2.1 Propriétés	359
2.2 Interfaces	365
2.3 Amis	366
2.4 Attributs	367
2.5 Méthodes	368
2.6 Événements	368
2.7 Types	370
2.8 Alias	371
3. Les méthodes	371
3.1 Paramètre	374
3.2 Exceptions	375
3.3 Code Source	377
3.4 Constructeur	378
3.5 Constructeur de classe	381
4. Création d'une classe	382
5. Exercice	387

Chapitre 10

Création d'un programme ABAP

1. Écran de sélection	389
1.1 Les champs	391
1.1.1 Sélection multiple	391
1.1.2 Sélection unique	397
1.2 Organisation	403
1.3 Les événements	405
1.4 Les variantes	408
2. Rapport ALV	416
2.1 La demande	416
2.2 L'analyse	417
2.3 L'algorithme	418

- 2.4 Le développement 418
 - 2.4.1 Écran de sélection 419
 - 2.4.2 Sélection des données 421
 - 2.4.3 Construction du rapport ALV 423
- 2.5 Tests techniques 429
- 3. Organisation finale du programme 429
- 4. Exercice 444

Chapitre 11
Corrigé des exercices

- 1. Corrigé chapitre Premiers pas sur SAP 447
- 2. Corrigé chapitre Instructions basiques ABAP 451
 - 2.1 La demande 451
 - 2.2 L'analyse 452
 - 2.3 L'algorithme 452
 - 2.4 Développement 453
 - 2.5 Tests unitaires 454
- 3. Corrigé chapitre Dictionnaire de données (DDIC) 455
- 4. Corrigé chapitre Les tables internes 459
 - 4.1 La demande 459
 - 4.2 L'analyse 460
 - 4.3 L'algorithme 460
 - 4.4 Développement 461
 - 4.5 Tests unitaires 465
- 5. Corrigé chapitre Les requêtes SQL 466
 - 5.1 La demande 466
 - 5.2 L'analyse 466
 - 5.3 L'algorithme 467
 - 5.4 Développement 468
 - 5.5 Tests unitaires 471
- 6. Corrigé chapitre Les fonctions 472
 - 6.1 La demande 472
 - 6.2 L'analyse 472
 - 6.3 L'algorithme 473

6.4	Développement	474
6.5	Tests unitaires	478
7.	Corrigé chapitre Les classes	479
8.	Corrigé chapitre Création d'un programme ABAP	484
8.1	La demande	484
8.2	L'analyse	484
8.3	L'algorithme	485
8.4	Le développement	486
8.5	Tests techniques	493
	Index	495



Chapitre 4

Instructions basiques ABAP

1. Variables et constantes

Comme dans tout langage de programmation, l'ABAP contient des variables et des constantes. Par définition, une variable est un symbole informatique associant un nom à une valeur qui peut varier durant l'exécution du programme. Cette définition s'applique également à une constante, à la différence près que sa valeur est fixée dès le début et ne changera jamais au cours de l'exécution du programme.

On retrouve sur SAP les types de variables suivants :

Types de variables ABAP

Type	Description	Longueur par défaut	Valeur par défaut	Exemple
c	Chaîne de caractères alphanumériques	1	"	'ABC012'
n	Numérique	1	0	5
d	Date	8	00000000	20090412
t	Heure	6	000000	134523
x	Hexadécimal	1	X'0'	65AF
i	Entier	4	0	5
p	Nombre à virgule	8	0	5,6
f	Format scientifique	8	0	2,2 E+209

Type	Description	Longueur par défaut	Valeur par défaut	Exemple
string	Texte long	Variable	"	N'importe quelle chaîne de caractères
xstring	String d'hexadécimal	Variable		N'importe quelle chaîne hexadécimale

■ Remarque

Le format date est de type AnnéeMoisJour (AAAAMMJJ), pour un affichage plus adéquat, il faudra toujours modifier la variable `date` (voir un exemple dans la section Opérations sur variable texte de ce chapitre).

Si le type entier **i** et le type numérique **n** sont comparés, il apparaît qu'ils sont sensiblement les mêmes. En fait, pour apporter un peu plus de précisions, le type entier **i** est, comme son nom l'indique, une chaîne numérique de nombres entiers alors que le type numérique **n** est aussi une chaîne numérique mais stockée sous forme de caractères, ce qui est pratique lors d'un travail avec des instructions sur des variables texte comme le **CONCATENATE** (cf. section Opérations sur variable texte de ce chapitre).

Maintenant que les différents types de variables ont été vus, il reste à savoir comment elles sont déclarées en ABAP. Pour cela, on utilise l'instruction **DATA** puis le nom de la variable (libre), sa longueur entre parenthèses, l'instruction de référence (qui peut être égale à **TYPE**, **LIKE**... comme dans l'exemple ci-dessous) puis son type :

```
DATA v_name(10) TYPE c.
DATA v_date    TYPE d.
DATA v_hour    TYPE t.
DATA v_year(4) TYPE n.
```

Pour que ce soit plus lisible, l'instruction **DATA** peut être suivie de deux points ':' pour ajouter plusieurs paramètres à sa composante. Ainsi, il apparaîtra :

```
DATA: v_name(20) TYPE c,
      v_date    TYPE d,
      v_hour    TYPE t,
      v_year(4) TYPE n.
```

Remarque

Chaque fin de ligne se termine par une virgule (,) afin de séparer les paramètres composant le **DATA** mais cela ne signifie pas que ce soit la fin de l'instruction. Comme déjà évoqué dans le chapitre Premiers pas sur SAP - Premier programme ABAP, la fin d'une instruction en ABAP se termine toujours par un point (.).

Quatre variables ont été créées : la variable `V_NAME` de type chaîne de caractères avec une longueur de dix positions, `V_DATE` de type date et donc d'une longueur fixe de huit positions, `V_HOUR` de type heure et donc aussi avec une longueur fixe mais de six positions, et enfin la variable `V_YEAR` de type numérique avec une longueur de huit positions.

Remarque

La nomenclature choisie ici commence par `v_` mais est totalement libre. D'un point de vue personnel, les variables utilisées commencent toujours par `gv_` pour des variables globales, `lv_` pour celles en locales, `pv_` pour une variable en paramètre... Ainsi un simple coup d'œil sur le nom indique très rapidement à quel niveau la variable a été déclarée.

Comme cité un peu plus haut, l'instruction de référence peut être égale à **TYPE** ou **LIKE**. Pour comprendre la différence entre les deux, voici un exemple :

```
DATA: v_name(10) TYPE c,  
      v_name2    LIKE v_name.
```

La variable `V_NAME` est déclarée avec un type chaîne de caractères et de longueur 10 et `V_NAME2` quant à elle prend comme référence la variable `V_NAME`. Ainsi, **TYPE** va pointer directement vers un type spécifique alors que le **LIKE** va en prendre indirectement une référence. Dans une programmation objet (cf. chapitre Les classes), le **TYPE** est à privilégier.

Concernant les constantes, c'est exactement le même principe sauf que l'instruction commencera par **CONSTANTS** et devra comporter obligatoirement une valeur fixe avec **VALUE**.

```
CONSTANTS : c_yes(1)    TYPE c VALUE 'X',  
            c_answer(2) TYPE i VALUE '42',  
            c_pi        TYPE p DECIMALS 2 VALUE '3.14'.
```

Trois constantes ont été déclarées : la première `C_YES` de type chaîne de caractères alphanumériques de longueur d'une seule position aura comme valeur 'X', la deuxième `C_ANSWER` est un type entier d'une longueur de deux positions et d'une valeur de '42', et enfin la dernière `C_PI` est un nombre avec virgule d'une valeur de '3,14'. Il est à noter qu'en ABAP, le point (.) est utilisé pour les décimales.

Remarque

Le type *p* doit être accompagné par l'instruction **DECIMALS** qui va définir le nombre de chiffres après la virgule, sinon la variable associée sera considérée comme un nombre entier.

Il existe aussi des variables système en arrière-plan, accessibles par tous les programmes, et dans les mandants, stockées dans la table **SY**.

Elles informent du statut global du système comme :

Le mandant de connexion	SY-MANDT
Le nom de l'utilisateur	SY-UNAME
La date	SY-DATUM
L'heure	SY-UZEIT

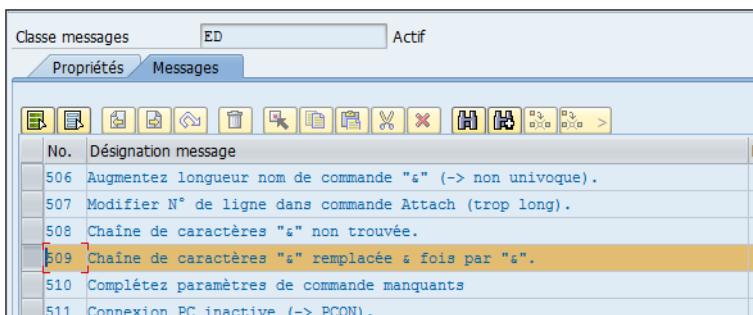
Ainsi que des informations plus spécifiques au programme exécuté, notamment :

Code retour d'un traitement	SY-SUBRC
Statut du message après exécution d'une opération spécifique	SY-MSGTY
Le numéro de la ligne d'une table lue lors d'une boucle sur table interne (cf. chapitre Les tables internes - Organisation et Lecture)	SY-TABIX
Le numéro d'un index utilisé également dans une boucle de type DO ou WHILE (voir un peu plus loin la section sur Les boucles)	SY-INDEX

Remarque

Le statut de message *SY-MSGTY* sera toujours accompagné d'une classe de message (*SY-MSGID*), et de son numéro (*SY-MSGNO*), avec les paramètres (*SY-MSGV1*, *SY-MSGV2*, *SY-MSGV3*, et *SY-MSGV4*) consultables via la transaction SE91.

Exemple avec la classe de message ED, numéro de message 509



Ce message comporte trois paramètres symbolisés par le caractère '&'. Ainsi la valeur du premier paramètre sera stockée dans la variable système SY-MSGV1, le deuxième dans SY-MSGV2, et le troisième dans SY-MSGV3 (le quatrième sera vide car non utilisé). Aussi, si les valeurs sont définies de la façon suivante :

- SY-MSGTY = 'I'
- SY-MSGID = 'ED'
- SY-MSGNO = '509'
- SY-MSGV1 = 'Hello'
- SY-MSGV2 = '3'
- SY-MSGV3 = 'Bye'

SAP aura toutes les informations pour construire un message d'information (I) comportant le texte : Chaîne de caractères "Hello" remplacée 3 fois par "Bye".

Pour illustrer ce chapitre un petit programme pour afficher quelques informations du système et une constante contenant la valeur 'Hello world'.

```
CONSTANTS : c_text(11)  TYPE c VALUE 'Hello World'.  
  
WRITE:/ 'Variable système ABAP'.  
WRITE:/ 'Mandant :      ', sy-mandt.  
WRITE:/ 'Utilisateur : ', sy-uname.  
WRITE:/ 'Date :         ', sy-datum.  
WRITE:/ 'Heure :         ', sy-uzzeit.  
WRITE / c_text.
```

Exemples for the chapter 4	
Exemples for the chapter 4	
Variable système ABAP	
Mandant :	300
Utilisateur :	USER
Date :	17.11.2017
Heure :	17:39:37
Hello World	

Remarque

Ici aussi, on peut utiliser les deux points (':') avec l'instruction **WRITE** qui contiendra ainsi une série de paramètres. Elle est suivie de la barre '/' signifiant une nouvelle ligne, puis d'un texte lui-même suivi d'une virgule, et enfin une variable. Ainsi, comme pour l'exemple du *DATA* en début de chapitre, il aurait été possible d'écrire ce code de la manière suivante :

```
CONSTANTS : c_text(11) TYPE c VALUE 'Hello World'.

WRITE:/ 'Variable système ABAP',
        / 'Mandant : ', sy-mandt,
        / 'Utilisateur : ', sy-uname,
        / 'Date : ', sy-datum,
        / 'Heure : ', sy-uzeit,
        / c_text.
```

2. Opérations arithmétiques

Comme dans tout langage de programmation, les variables de types numériques (entier, avec décimales...) peuvent être utilisées dans des opérations arithmétiques.

Tout d'abord, pour assigner une valeur à une variable, les instructions **MOVE** ou égal (=) sont utilisées.

Exemple

```
DATA: v_a(2) TYPE i,
      v_b(2) TYPE i,
      v_c(2) TYPE i,
      v_d(2) TYPE i.

v_a = 3.
v_b = v_a.
MOVE 5 TO v_c.
MOVE v_c TO v_d.

WRITE:/ 'Valeur de v_a : ', v_a,
        / 'Valeur de v_b : ', v_b,
        / 'Valeur de v_c : ', v_c,
        / 'Valeur de v_d : ', v_d.
```