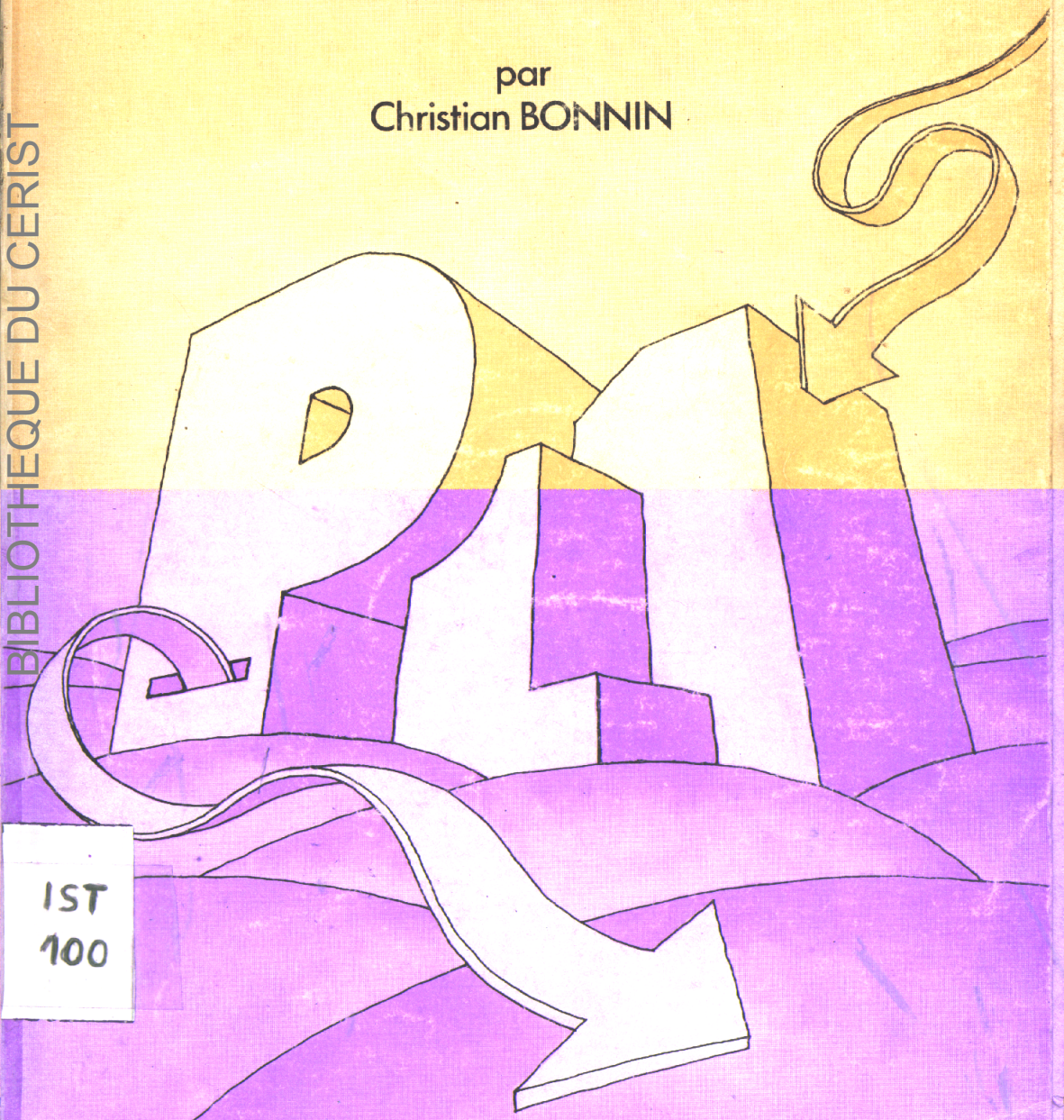
PRATIQUE DU PL/1 ET PROGRAMMATION STRUCTURÉE

par
Christian BONNIN



EDITIONS EYROLLES

PRATIQUE DU PL/1
ET
PROGRAMMATION STRUCTURÉE

par

Christian BONNIN

Ingénieur Informaticien I. D. N., Licencié ès Sciences
Chef de Systèmes et Applications Informatiques
à la Compagnie I. B. M. France

ÉDITIONS EYROLLES
61, boulevard Saint-Germain - PARIS-V^e^
1976

TABLE DES MATIÈRES

AVERTISSEMENT

Au début des années cinquante commencèrent à apparaître les langages symboliques destinés à faciliter la programmation des ordinateurs. Ces langages étaient le plus généralement du type ASSEMBLEUR, c'est-à-dire directement dérivés du langage machine.

En 1956 apparut FORTRAN, premier langage véritablement indépendant de la structure interne de l'ordinateur, mais réservé presque exclusivement aux programmeurs scientifiques.

Le succès remporté par FORTRAN fit des jaloux dans le domaine de la gestion. Parmi eux, le gouvernement des États-Unis qui commençait à rencontrer de sérieuses difficultés d'harmonisation des méthodes de gestion par suite de la diversité des matériels et langages utilisés au sein de ses diverses administrations, commanda une étude qui aboutit en 1959 au langage COBOL, strictement réservé aux applications de gestion.

Le monde informatique s'est ainsi vu séparer en deux sectes usant de langages incompatibles : FORTRAN incapable de traiter des fichiers de gestion et COBOL inapte à réaliser des calculs scientifiques.

Constatant les insuffisances de ces deux principaux langages, un groupe d'utilisateurs associé à IBM conçut en 1964 le PL/1 (Programming Language Number One), langage de programmation convenant aux applications de gestion aussi bien que scientifiques et qui permet aux programmeurs de traiter la totalité de leurs tâches de programmation, tout en maîtrisant presque complètement les possibilités de l'ordinateur.

Dix ans après, la guerre des langages continue, mais peu à peu PL/1 s'impose partout où les systèmes de traitement de l'information sont les plus évolués. PL/1 est conçu de telle manière qu'un programmeur, quelle que soit l'étendue de son expérience, puisse l'utiliser aisément à son propre niveau. Simple pour le débutant, car il n'est pas nécessaire que le programmeur connaisse tout PL/1 pour pouvoir l'utiliser, il devient un instrument puissant pour le programmeur expérimenté.

Le présent ouvrage n'étant pas une initiation à l'informatique et à la programmation, nous avertissons le lecteur qu'il suppose connus les principes de base de fonctionnement et programmation des ordinateurs.

D'autre part, ne sont reprises ici que les spécifications PL/1 d'usage courant, c'est-à-dire celles concernant les traitements purement séquentiels. Les extensions suivantes : TRI DYNAMIQUE, PRE-PROCESSEUR, ACCES DIRECT, MULTI TASKING feront l'objet d'un autre ouvrage : Techniques avancées de programmation PL/1.

Intentionnellement, cet ouvrage est orienté vers la programmation des ordinateurs IBM et se réfère au compilateur PL/1 optimiseur, communément appelé PL/I.

Rappelons qu'un programme PL/1 est organisé en blocs ou procédures, ce qui lui confère d'exceptionnelles aptitudes à la modularité.

Les instructions des procédures sont constituées de mots-clés en langue anglaise mais contrairement aux autres langages, ces mots-clés ne sont pas réservés, c'est-à-dire que le programmeur peut les utiliser comme noms symboliques pour désigner des zones de données.

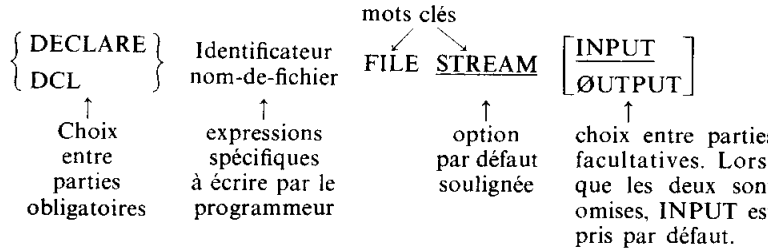
PL/1 offre de nombreuses options pour les instructions, les descriptions de fichiers ou de données. Chaque fois que le langage permet une alternative, si le programmeur n'a pas exprimé de choix, le compilateur adopte d'office l'une des possibilités, c'est l'option par défaut.

NOTATIONS UTILISÉES

Les notations utilisées pour décrire les expressions PL/1 sont les expressions couramment admises, c'est-à-dire :

. les mots-clés sont imprimés en lettres majuscules ; exemple READ ;
. les noms symboliques, identificateurs ou expressions spécifiques à écrire par le programmeur sont imprimés en caractères minuscules ;
. les expressions facultatives sont imprimées entre crochets [ ]. Lorsque l'on peut choisir entre plusieurs expressions facultatives, elles sont regroupées entre crochets ;
. lorsque l'on peut choisir entre plusieurs expressions mais que, au moins l'une d'elles est obligatoire, elles sont regroupées entre accolades { } ;
. les options par défaut sont soulignées STATIC.

Exemple de notations :



REMARQUE : théoriquement les normes officielles de programmation énoncent que la lettre Ø doit être écrite O et le zéro doit être barré Ø. En pratique, les programmeurs utilisent généralement la notation inverse lettre Ø et chiffre 0 ; c'est cette notation que nous adoptons.

CHAPITRE PREMIER

PRINCIPES DE BASE

Les instructions PL/1 sont généralement écrites sur des bordereaux de perforation de cartes en format libre, c'est-à-dire qu'une instruction PL/1 peut occuper une partie de carte ou plusieurs cartes.

Dans la pratique cependant, les programmeurs n'utilisent que les colonnes 2 à 72 pour les instructions et réservent la zone 73-80 pour un numérotage des cartes afin de faciliter le reclassement en cas d'incident (ceci est une option des compilateurs).

Les programmeurs chevronnés s'imposent aussi une présentation soignée en alignant les blocs d'instructions sur des colonnes identiques pour en faciliter la compréhension et surtout la maintenance.

La fin d'une instruction est marquée par le caractère point-virgule.

Il n'y a pas de mots réservés en PL/1.

Les caractères utilisables en PL/1

Le langage PL/1 comporte normalement 60 caractères, mais pour satisfaire aux contraintes de certains matériels, il existe un sous-ensemble à 48 caractères.

Les caractères utilisables sont les suivants :

Ensemble 60 caractères		Ensemble 48 caractères
0 à 9	les chiffres	0 à 9
A à Z	alphabet augmenté de 3 caractères ci-dessous	A à Z
$	Dollar traduit par F en France (il s'agit d'un F spécial)	$
@	arrobas	n'existe pas
#	numéro	n'existe pas

b	blanc ou espace	*b*
=	signe — d'égalité dans une comparaison ; exemple : si A = B faire... — d'affectation dans un calcul ; exemple : A = C + D signifie ajouter D à C et affecter le résultat à la zone A.	=
+	signe +	+
−	signe −	−
*	signe de multiplication ou astérisque	*
/	signe de division	/
(	parenthèse gauche	(
)	parenthèse droite	)
,	virgule	,
.	point de ponctuation ou point décimal — faire attention que pour les nombres, le rôle du point et de la virgule est inversé aux USA par rapport à la France, or PL/1 est américain.	.
'	guillemet ou apostrophe	'
%	pourcent	//
;	point-virgule	,.
:	deux-points	..
¬	symbole NON (opérateur logique)	NØT
&	symbole ET (opérateur logique ET exclusif)	AND
|	symbole OU (opérateur logique OU inclusif)	ØR
>	plus grand que	GT
<	plus petit que	LT
?	point d'interrogation	n'existe pas
_	blanc souligné (c'est le caractère qui sert à souligner un texte sur une machine à écrire).	n'existe pas

Certains caractères spéciaux peuvent être associés pour désigner des opérateurs supplémentaires :

**	élévation à la puissance	**
||	concaténation	CAT
−>	pointeur	PT

L'association de deux opérateurs complémentaires est aussi possible :

Exemple : < = signifie plus petit ou égal
> = » plus grand ou égal
¬ = » non égal
¬ < » non inférieur
¬ > » non supérieur

REMARQUE : le sous-ensemble 48 caractères étant moins fréquemment utilisé, nous nous bornerons, dans la suite de cet ouvrage, à l'ensemble de 60 caractères pour ne pas compliquer inutilement les exposés.

Les commentaires

Des commentaires, ignorés par le compilateur, peuvent être introduits dans un programme PL/1 pour en expliquer la logique.

Les couples de caractères /* et */ doivent entourer ces commentaires. Par exemple

```
/* LES COMMENTAIRES SONT INDISPENSABLES A */
/* LA COMPREHENSION D'UN PROGRAMME */.
```

L'ensemble des 60 caractères du langage peut être utilisé dans le commentaire et, bien sûr, les colonnes 1 et 73 à 80 de la carte ne peuvent être utilisées (sauf options contraires prises à l'installation du compilateur).

LES IDENTIFICATEURS

Les identificateurs sont des noms symboliques servant à désigner une zone de donnée, une adresse symbolique, un fichier...

Écriture

Les identificateurs comportent de 1 à 31 caractères choisis parmi les suivants :

les chiffres 0 à 9
l'alphabet A à Z et $ @ #
le blanc souligné _

Exemples :

CØDE_ARTIC, MATRIC1, TØTAL_GEN

REMARQUE : les caractères @ # _ n'existant pas dans le jeu à 48 caractères, nous déconseillons leur emploi. De même pour $ car on confond trop facilement le F de franc avec la lettre F en France.

Restrictions

— Un identificateur doit commencer par un caractère alphabétique.

— Un identificateur doit toujours être suivi et précédé par un caractère blanc ou un délimiteur (caractère spécial, ou parenthèses).

— Un identificateur ne peut excéder 8 caractères qu'à la condition de le scinder en blocs de huit caractères au plus séparés par des blancs soulignés.

Qualification

Il peut arriver qu'un même identificateur soit utilisé pour désigner deux zones de données différentes, par exemple INDICAT d'un fichier-1 et INDICAT d'un fichier-2. On distingue ces deux zones de données en qualifiant l'identificateur par un préfixe FICH1.INDICAT et FICH2.INDICAT.

Mot clé

Les mots-clés sont des identificateurs, bâtis à partir de mots de la langue anglaise, et qui constituent les codes opérations. Mais, contrairement à d'autres langages, ces mots ne sont pas des mots réservés c'est-à-dire qu'on peut très bien les utiliser comme nom-données symboliques, labels ou autres.

Exemple, dans l'expression IF A > B THEN A = B + C les identificateurs IF et THEN sont des mots-clés.

Nom-Externe

Un nom-Externe est un identificateur destiné à être communiqué à l'extérieur du programme, c'est-à-dire à un autre module du système d'exploitation.

Un Nom-Externe est limité à 7 caractères et ne peut contenir de blanc souligné.

REMARQUE : si un Nom-Externe comporte plus de 7 caractères, le compilateur ne retient que les 4 caractères de gauche et les 3 de droite.

Les labels

Les adresses symboliques d'un programme PL/1 sont appelées labels. Ils s'écrivent de la manière suivante : label : instruction ;

Exemple :

```
DEBUT : A = B + C;
```

On se débranche à un point particulier d'un programme en lui affectant un label et en écrivant l'instruction GØ TØ label ;

Exemple :

```
GØ TØ DEBUT;
```