

Daniel Thalmann
Bernard Levrat

CONCEPTION ET IMPLANTATION DE LANGAGES DE PROGRAMMATION:

**Une introduction à
la compilation**

BIBLIOTHEQUE DU CERIST



MICHEL CHARVILLAT



**gaëtan morin
éditeur**

COLLECTION

SYSTÈMES D'INFORMATION

dirigée par LIN GINGRAS

professeur à l'Université Laval

Daniel Thalmann
Bernard Levrat

CONCEPTION ET IMPLANTATION

DE LANGAGES

DE PROGRAMMATION:

**Une introduction à
la compilation**



gaëtan morin
éditeur

C.P. 965, CHICOUTIMI, P.Q. TEL.: 545-3333

ISBN 0-88612-020-9

TABLEAU: (Couverture) Oeuvre de Michel
Champagne.

Imprimerie: La Librairie Commerciale Ltée

Dépôt légal: 1er trimestre 1979
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada.

TOUS DROITS RÉSERVÉS
Copyright © 1979. Gaëtan Morin & Associés Ltée

N^o INU.
2573

AVANT-PROPOS



Ce volume consacré à la compilation devrait permettre, parallèlement à un exposé méthodologique, d'aboutir à l'écriture d'un compilateur par les étudiants, afin de mettre en oeuvre une partie des techniques présentées.

Pourtant, la compilation, science de la traduction automatique de langages de programmation, présente des liens certains avec la théorie des langages et un résumé de cette théorie est présenté au début de l'ouvrage.

Pour chaque sujet, les bases théoriques sont exposées, mais la plus large part est laissée aux algorithmes.

Ceux-ci sont formalisés en langage Pascal, certains ont été traduits d'Algol 60 ou de PL/1, les autres sont originaux.

Dans certains de ces algorithmes, l'accent a été davantage mis sur la clarté que sur l'efficacité.

Le Pascal utilisé est celui décrit dans "Pascal, User Manual and Report" de K. Jensen et N. Wirth. Les exemples sont empruntés aux langages les plus courants, à savoir : Algol 60, Fortran, PL/1, Cobol, Pascal, Basic, etc.

Les thèmes abordés dans ce cours sont les éléments fondamentaux de la compilation : l'analyse lexicale, l'analyse syntaxique, la génération de code, l'optimisation et le traitement des erreurs.

Les méthodes d'analyse syntaxique exposées ici sont groupées selon les 2 courants : méthodes descendantes et méthodes ascendantes.

On sait que dans tout compilateur, une table des symboles est indispensable, les différentes techniques d'accès et les problèmes posés par les variables locales sont également abordés.

Pour décrire la génération de code, un langage d'assemblage pour une machine hypothétique est introduit et la compilation des instructions de contrôle est expliquée à l'aide de cet assembleur.

Deux concepts fondamentaux de tout langage sont décrits dans ce cours : les structures de données et les sous-programmes, et les difficultés que pose leur implantation. Ces 2 concepts font apparaître la nécessité de présenter certaines techniques de gestion de la mémoire.

Les liens existants entre l'analyse syntaxique et la génération de code étant particulièrement intéressants dans le cadre du traitement des expressions arithmétiques, cet aspect est développé.

Une large place est faite à l'optimisation du code généré, divers algorithmes sont présentés; une distribution entre optimisations dépendantes et indépendantes de la machine a paru judicieuse; les dangers des optimisations sont aussi évoqués.

Le traitement des erreurs à tous les niveaux, orthographe, syntaxe, sémantique, exécution est développé.

Les techniques de récupération et de réparation d'erreurs sont illustrées par de nombreux exemples. Un chapitre est consacré à la construction des interpréteurs et des simulateurs dont les liens avec les compilateurs sont évidents. L'évolution vers les compilateurs écrits en langage de haut niveau a conduit à considérer la portabilité de ceux-ci et la technique de "bootstrapping" est expliquée ici, tandis qu'un aperçu du concept de machine abstraite est présenté.

Si l'aboutissement de ce cours doit être la réalisation par les étudiants d'un compilateur pour un petit langage de type algorithmique, utilisant certaines techniques présentées, il est nécessaire d'intégrer un certain nombre d'exercices basés sur les autres techniques; un petit choix d'exercices figure donc à la fin de chaque chapitre, ainsi que des exercices de révision à la fin du volume.

Enfin, on a essayé de donner une bibliographie assez complète d'un sujet très vaste pour lequel ce volume ne représente qu'une introduction.

Les auteurs remercient Madame Liliane Noël et Monsieur Pierre-Alain Marti pour la création de l'ouvrage à partir du manuscrit.

BIBLIOTHEQUE DU CERIST

Table des matières

Page

1. <u>Théorie des langages</u>	
1.1 Définitions de base	1
1.2 Arbres de dérivation	3
1.3 Ambiguïtés	5
1.4 Productions particulières	7
1.5 L'analyse des chaînes	8
1.6 La structure d'un compilateur	8
2. <u>L'analyse lexicale</u>	
2.1 L'analyseur lexical	13
2.2 Exemple d'analyseur lexical	14
2.3 Les mots-clés du langage	16
3. <u>L'analyse syntaxique</u>	
3.1 Conceptions descendante et ascendante	21
3.2 Définition d'un langage	23
4. <u>Techniques ascendantes</u>	
4.1 La précedence d'opérateurs	29
4.2 La simple précedence	34
4.3 Les matrices de transition	42
4.4 Les analyseurs LR(K)	45
5. <u>Techniques descendantes</u>	
5.1 Les analyseurs LL(K)	53
5.2 La descente récursive	58
6. <u>La table des symboles</u>	
6.1 Généralités	65
6.2 Accès à la table	65
6.3 Variables locales	72
6.4 Caractéristiques d'un élément de la table des symboles	75



7.	<u>La génération de code</u>	
7.1	Type de code généré	79
7.2	Introduction d'un assembleur pour une machine hypothétique	79
8.	<u>La compilation des instructions de contrôle</u>	
8.1	Langage d'assemblage	85
8.2	Les instructions sélectives	86
8.3	Les instructions répétitives	90
8.4	Procédure de compilation d'une instruction de contrôle	93
9.	<u>La gestion de la mémoire</u>	
9.1	Généralités	97
9.2	Les procédures de base	99
9.3	"Ramasseur de miettes" (Garbage Collector)	108
10.	<u>L'implantation des structures de données</u>	
10.1	Généralités	111
10.2	Les vecteurs	112
10.3	Les matrices	112
10.4	Généralisation : tableaux multi-dimensions	114
10.5	Les chaînes	118
10.6	Les enregistrements	120
11.	<u>Les sous-programmes</u>	
11.1	Généralités	125
11.2	La transmission des paramètres	125
11.3	L'accès aux variables dans un sous-programme. Cas du Fortran, du Basic et du Cobol.	131
11.4	La récursivité et la structure de blocs. Cas de l'Algol.	136
11.5	Les procédures et les fonctions de Pascal	144

12. <u>Le traitement des expressions arithmétiques</u>	
12.1 Traitement par table de précedence d'opérateurs	155
12.2 Traitement par notation polonaise postfixée	157
12.3 Traitement par arbre	160
12.4 Traitement par la descente récursive	160
13. <u>L'optimisation</u>	
13.1 Généralités	169
13.2 Les optimisations indépendantes de la machine	169
13.3 Dangers de ces optimisations	172
13.4 Les optimisations dépendantes de la machine	173
13.5 Les algorithmes d'Anderson et de Nakata	176
13.6 Les théories de Sethi et Ullman	179
13.7 Application de la théorie des graphes	188
14. <u>Le traitement des erreurs</u>	
14.1 Généralités	197
14.2 Les classes d'erreurs	197
14.3 Les diagnostics	199
14.4 La récupération des erreurs	200
15. <u>Les interpréteurs et les simulateurs</u>	
15.1 Les interpréteurs	207
15.2 Les simulateurs de machines	208
16. <u>La portabilité des compilateurs et les machines abstraites</u>	
16.1 La portabilité	215
16.2 Le "bootstrapping"	216
16.3 Les machines abstraites	219
Exercices de révision	221
Bibliographie	225