C 4369

collection langages et algorithmes de l'informatique

sous la direction de JEAN VIGNES

Professeur à l'Université Paris VI

Conseiller scientifique à l'Institut Français du Pétrole

JEAN VIGNES

Professeur à l'Université Paris VI

Professeur à l'École Nationale Supérieure du Pétrole et des Moteurs

CLAUDE COURTY-LAJEUNESSE

Ingénieur E. P. F.

Ingénieur à la Compagnie Française d'Études et de Construction Technip

CLAUDE DEBOST-BENTZ

Ingénieur E. P. F.

Chargée de mission à la Société d'Études pour le Développement Économique et Social

Préface de

WLADIMIR MERCOUROFF

Chargé de mission à l'Informatique au Ministère de l'Éducation Nationale

ARRIVÉ LE
04 AVR. 1982
CENTRE ... IFIQUE
ET TECH... SFERTS
B.P. 315 ... GARE

BIBLIOTHEQUE DU CERIST

théorie et pratique de la programmation COBOL

1972

ÉDITIONS TECHNIP • 27 RUE GINOUX • 75737 PARIS CEDEX 15

technip

PRÉFACE

La deuxième moitié des années 50 a vu, aux Etats-Unis notamment, le développement explosif de l'industrie des ordinateurs et de l'utilisation de ceux-ci en matière de gestion.

Du côté scientifique, l'IBM avait considérablement facilité l'emploi des machines par la mise au point de FORTRAN, qui permettait - et qui permet toujours - aux utilisateurs de mettre au point très rapidement leurs programmes.

Mais le gros des forces informatiques est en gestion (de 70 à 80%), où l'on manipule de très grandes masses de données, sans toutefois leur faire subir de traitement aussi sophistiqué que dans les applications scientifiques.

Aussi la plupart des constructeurs ont-ils rapidement mis au point des langages spécialisés de gestion. Vers la fin des années 50, il y en avait trois : FLOW-MATIC (Univac), COMTRAN (IBM) et FACT (Minneapolis-Honeywell). Ils étaient cependant assez différents et en tous cas non transférables d'un constructeur à l'autre.

Pour rémédier à cette diversité, cause fâcheuse de pertes d'énergie, le principal utilisateur d'ordinateurs en gestion, le gouvernement américain (et surtout le Pentagone) a imposé en 1959 la définition d'un langage de gestion commun (Common Business Oriented Langage) ou COBOL.

En 1960 presque tous les constructeurs travaillaient à cette définition et à la réalisation de compilateurs. La normalisation est intervenue en 1963. En outre, COBOL est devenu, depuis 1968, un standard américain ("USASI X3. 23. 1968"). Ceci signifie que tous les systèmes classiques (à l'exception de ceux destinés au Process-Control, par exemple) doivent être munis d'un compilateur COBOL.

Ceci signifie donc aussi que tous les programmeurs de gestion doivent connaître le COBOL.

Le langage est un très bon outil de gestion classique. Ses spécifications d'entrées-sorties sont excellentes. Il est disponible chez tous les constructeurs et les compilateurs sont au point, en général. Mais il souffre d'un certain nombre de défauts (verbosité, absence de

BIBLIOTHEQUE DU CERIST

passage d'arguments, inaptitude au traitement des listes, traitement des chaînes fort malcommode, etc.). On peut donc encore rêver d'un langage standardisé plus souple, plus riche et plus moderne...

L'ouvrage de J. VIGNES, Mme COURTY et Mme DEBOST présente COBOL de façon très claire. Une première partie donne les notions indispensables à la bonne intelligence du langage proprement dit, qui rend le livre très pédagogique. On y trouve une description des codifications internes et surtout des fichiers et des opérations dont ils sont l'objet, raison d'être de COBOL.

Il est bon que ce livre paraisse, comblant un vide en littérature française, d'autant qu'il donne les différences entre les différentes réalisations.

Ceci étant, il ne faudrait pas que la connaissance de COBOL bloque le programmeur dans une sorte de contentement de soi, eu égard à l'officialisation du langage, et qu'il se préoccupe de ce seul outil de codage. S'il désire rester "dans la course", COBOL n'est qu'un outil - particulièrement important -, parmi d'autres dont il doit apprendre à jouer.

W. MERCOUROFF
Chargé de mission à l'Informatique au
Ministère de l'Education Nationale

BIBLIOTHEQUE DU CERIST

TABLE DES MATIÈRES

PREFACE V
HISTORIQUE VII
AVERTISSEMENT XXI

I. INTRODUCTION

I.1. LA NOTION DE CERVEAUX ELECTRONIQUES 1

I.2. LES ORDINATEURS ET LEURS LANGAGES 2

II. GENERALITES

II.1. LE BORDEREAU DE PERFORATION ET LA CARTE PERFOREE 11

II.2. CARACTERES UTILISABLES EN COBOL 14

II.3. PERFORATION DES CARACTERES SUR LA CARTE 15

II.4. CODAGE BINAIRE DES CARACTERES EN MACHINE 15

II.4.1. Tableau des codifications CDC de la série des 6000 18

II.4.2. Tableau des codifications IBM de la série des 360 et CII de la série IRIS 50 20

II.4.3. Tableau des codifications UNIVAC de la série 1108 22

II.4.4. Tableau des codifications HONEYWELL BULL de la série GE 600 24

BIBLIOTHEQUE DU CERIST

II.5. CODAGE DES CARACTERES SUR LES BANDES MAGNETIQUES 26

II.5.1. Les bandes magnétiques 26

II.5.1.1. Description physique 26
II.5.1.2. Représentation des caractères sur la bande ... 28

II.5.2. Code DCB 7 pistes 29

II.5.3. Code des bandes à 9 pistes 32

II.6. REPRESENTATION DES NOMBRES QUELCONQUES EN MEMOIRE CENTRALE DE L'ORDINATEUR 33

II.6.1. Représentation des nombres par caractères 34

II.6.1.1. Nombres non signés 34
II.6.1.2. Nombres signés 35
II.6.1.3. Représentation des nombres en mode hexadecimal condensé 37

II.6.2. Représentation des nombres par leur valeur numérique ... 39

II.6.2.1. Représentation en mode entier 39
II.6.2.2. Représentation en mode flottant 40

II.7. MODES D'EXECUTION DES OPERATIONS ARITHMETIQUES 42

II.7.1. Arithmétique entre nombres représentés en caractères ... 43

II.7.1.1. L'addition et la soustraction 43
II.7.1.2. La multiplication et la division 46

II.7.2. Arithmétique entre nombres représentés par leur valeur numérique 46

II.7.2.1. Opérations en mode entier 47
II.7.2.2. Opérations en mode flottant 48

BIBLIOTHEQUE DU CERIST

III. PREPARATION ET ORGANISATION D'UN PROGRAMME COBOL

III.1. LES FICHIERS 51

III.2. LES ENREGISTREMENTS 52

III.2.1. L'enregistrement logique 52

III.2.2. L'enregistrement physique 54

III.2.3. Les enregistrements Label 64

III.3. LES DIVERSES ORGANISATIONS DES FICHIERS ET LEURS TECHNIQUES D'ACCES 66

III.3.1. Les diverses organisations 66

III.3.1.1. Organisation séquentielle 66
III.3.1.2. Organisation séquentielle indexée 67
III.3.1.3. Organisation à accès direct 67

III.3.2. La création des fichiers 67

III.3.2.1. Création d'un fichier à organisation séquentielle 67
III.3.2.2. Création d'un fichier à organisation séquentielle indexée 67
III.3.2.3. Création d'un fichier à organisation à accès direct 70

III.3.3. Consultations des fichiers : techniques d'accès aux fichiers 71

III.3.3.1. Accès à un fichier à organisation séquentielle 71
III.3.3.2. Accès à un fichier à organisation séquentielle indexée 72
III.3.3.3. Accès à un fichier à organisation à accès direct 72

III.4. LES ORGANIGRAMMES 73

III.4.1. Symboles pour organigrammes des données 74

III.4.1.1. Symboles d'information 74
III.4.1.2. Symboles de traitement 75
III.4.1.3. Lignes de liaison 76
III.4.1.4. Symboles auxiliaires 77

BIBLIOTHEQUE DU CERIST

III.4.2. Symboles pour organigrammes de programmation 77

III.4.2.1. Symboles de traitement 77
III.4.2.2. Symboles logiques 78
III.4.2.3. Symboles auxiliaires 78

III.5. EXEMPLE : FACTURATION DES COMMUNICATIONS TELEPHONIQUES DES ABONNES D'UNE VILLE 78

III.5.1. Constitution de fichiers 78

III.5.1.1. Fichiers d'entrée 79
III.5.1.2. Fichier de sortie 79

III.5.2. Elaboration de l'organigramme 80

IV. GENERALITES COBOL

IV.1. LE LANGAGE COBOL 89

IV.2. CARACTERES UTILISES POUR L'ECRITURE D'UN PROGRAMME 90

IV.3. STRUCTURE D'UN PROGRAMME 91

IV.3.1. Identification Division 92

IV.3.2. Environment Division 92

IV.3.3. Data Division 92

IV.3.4. Procedure Division 93

IV.4. TYPES DE MOTS UTILISES 93

IV.4.1. Les noms 93

IV.4.1.1. Nom-donnée 94
IV.4.1.2. Nom-procédure 95
IV.4.1.3. Nom-condition 96
IV.4.1.4. Nom-fichier 97

BIBLIOTHEQUE DU CERIST

IV.4.1.5. Nom-hardware ou non-unité 98
IV.4.1.6. Nom-bibliothèque 101
IV.4.1.7. Nom-qualifié 102
IV.4.1.8. Nom-indicé 103

IV.4.2. Les constantes 106

IV.4.2.1. Les littéraux numériques 107
IV.4.2.2. Les littéraux non-numériques 108
IV.4.2.3. Les constantes nommées 110
IV.4.2.4. Les constantes figuratives 111

IV.4.3. Les verbes 113

IV.4.3.1. Les verbes relatifs au programme 113
IV.4.3.2. Les verbes d'information pour le compilateur. 116
IV.4.3.3. Les verbes relatifs au tri 116
IV.4.3.4. Le verbe particulier USE 116

IV.4.4. Les mots facultatifs 117

IV.4.5. Les mots réservés 118

IV.5. LES EXPRESSIONS 124

IV.5.1. Expressions arithmétiques 124

IV.5.1.1. Opérateurs arithmétiques 124
IV.5.1.2. Règles d'écriture et exemples 125
IV.5.1.3. Hiérarchie des opérations arithmétiques 125
IV.5.1.4. Utilisation des parenthèses 126

IV.5.2. Expressions logiques 127

IV.5.2.1. Opérateurs de relation 127
IV.5.2.2. Opérateurs logiques 129
IV.5.2.3. Hiérarchie des opérateurs logiques 130
IV.5.2.4. Ecritures particulières d'expressions logiques 130

IV.6. LA PONCTUATION 134

IV.7. SIGNIFICATION DU SIGNE = EN COBOL 135

EXERCICES DU CHAPITRE IV 136

BIBLIOTHEQUE DU CERIST

V. IDENTIFICATION DIVISION

V.1. EXPRESSIONS OBLIGATOIRES 142

V.2. EXPRESSIONS OPTIONNELLES 143

V.3. EXEMPLE RECAPITULATIF D'ECRITURE D'UNE IDENTIFICATION DIVISION .. 144

VI. ENVIRONMENT DIVISION

VI.1. CØNFIGURATIØN SECTIØN 146

VI.1.1. SØURCE-CØMPUTER. *nom-d'ordinateur* 146

VI.1.2. ØBJECT-CØMPUTER. *nom-d'ordinateur* 146

VI.1.3. SPECIAL-NAMES. *clauses* 146

VI.1.3.1. *littéral-non-numérique*$_1$ IS *mnémonique*$_1$ 147
VI.1.3.2. *nom-hardware*$_1$ IS *mnémonique*$_2$ 147
VI.1.3.3. CURRENCY SIGN IS *littéral* 148
VI.1.3.4. DECIMAL-PØINT IS CØMMA 148

VI.2. INPUT-ØUTPUT SECTIØN 149

VI.2.1. FILE-CØNTRØL 149

VI.2.1.1. SELECT 150
VI.2.1.2. ØPTIØNAL 151
VI.2.1.3. RENAMING 151
VI.2.1.4. ASSIGN 151
VI.2.1.5. MULTIPLE REEL 151
VI.2.1.6. RESERVE {NØ / *entier*} ALTERNATE {AREA / AREAS} 152
VI.2.1.7. ACCESS MØDE IS SEQUENTIAL 152
VI.2.1.8. ACCESS MØDE IS RANDØM 152
VI.2.1.9. ØRGANIZATIØN MØDE IS *option* 153
VI.2.1.10. *option* KEY 153

VI.2.2. I-Ø-CØNTRØL 153

VI.3. EXEMPLE RECAPITULATIF D'ECRITURE D'UNE ENVIRØNMENT DIVISIØN 154

VI.3.1. Explications de l'exemple CDC 6600 159

VI.3.2. Explications de l'exemple IBM 360/50 164

BIBLIOTHEQUE DU CERIST

VII. DATA DIVISION

VII.1. ORGANISATION DE L'INFORMATION 169

VII.1.1. Les niveaux 169

VII.1.1.1. Notion d'élément 169
VII.1.1.2. Notion de sous-groupe 171
VII.1.1.3. Notion de groupe 172
VII.1.1.4. Règles d'écriture des nombres-niveau 174
VII.1.1.5. Conseils au programmeur 175

VII.1.2. Les niveaux spéciaux (66, 77, 88) 175

VII.1.2.1. Nombre-niveau 66 : nombre-niveau de regroupement 176
VII.1.2.2. Nombre-niveau 77 : niveau d'éléments indépendants 177
VII.1.2.3. Nombre-niveau 88 : nombre niveau de définition d'un nom-condition 178

VII.2. ORGANISATION DE LA DATA DIVISION 178

VII.2.1. FILE SECTIØN 180

VII.2.1.1. RECØRDING MØDE IS *option* 181

VII.2.1.2. BLØCK CØNTAINS *entier-1* {RECØRDS / CHARACTERS} 183

VII.2.1.3. RECØRD CØNTAINS[*entier-2* TØ] *entier-3* CHARACTERS 184

VII.2.1.4. LABEL{RECØRDS ARE / RECØRD IS} {STANDARD / ØMITTED / *nom-donnée*$_1$} 184

VII.2.1.5. DATA {RECØRDS ARE / RECØRD IS} *nom-donnée*$_2$[*nom-donnée*$_3$] 185

VII.2.1.6. Exemple d'écriture du niveau FD 187
VII.2.1.7. Description des enregistrements logiques d'un fichier 189
VII.2.1.8. Description des enregistrements logiques d'un fichier de sortie 212

VII.2.2. WØRKING-STØRAGE SECTIØN 224

VII.2.2.1. *nom-donnée1* REDEFINES *nom-donnée*$_2$ 226

VII.2.2.2. SYNCHRØNIZED {LEFT / RIGHT} 226

VII.2.2.3. VALUE 229

BIBLIOTHEQUE DU CERIST

VII.2.3. CØNSTANT SECTIØN 232

VII.2.4. CØMMØN-STØRAGE SECTIØN, LINKAGE SECTIØN 232

EXERCICES DU CHAPITRE VII 241

VIII. PROCEDURE DIVISION

VIII.1. GENERALITES 251

VIII.2. ORDRE DE TRAITEMENT DES INFORMATIONS 252

VIII.2.1. MØVE 252

VIII.3. LES ORDRES DE CALCUL ARITHMETIQUE 259

VIII.3.1. ØN SIZE ERRØR *instructions.* 260

VIII.3.2. RØUNDED 261

VIII.3.3. ADD 262

VIII.3.4. SUBTRACT 264

VIII.3.5. DIVIDE 266

VIII.3.6. MULTIPLY 268

VIII.3.7. CØMPUTE 269

VIII.4. LES ORDRES DE CONTROLE 272

VIII.4.1. Ordre de comparaison 272

VIII.4.2. Ordres de débranchement 276

VIII.4.2.1. Le débranchement inconditionnel 277
VIII.4.2.2. Le débranchement conditionnel 277
VIII.4.2.3. Ordre modifiant les débranchements 278
VIII.4.2.4. Ordres de débranchement et retour 279
VIII.4.2.5. EXIT 290

VIII.4.3. L'ordre complémentaire STØP 292

BIBLIOTHEQUE DU CERIST

VIII.5. LES ORDRES D'ENTREE-SORTIE 293

VIII.5.1. Généralités 293

VIII.5.2. ACCEPT 294

VIII.5.3. DISPLAY 296

VIII.5.4. ØPEN 297

VIII.5.5. READ 299

VIII.5.6. WRITE 301

VIII.5.6.1. La forme 1. 302
VIII.5.6.2. La forme 2. 304

VIII.5. 7. CLØSE 304

VIII.5. 8. EXAMINE 306

VIII.5. 9. NØTE 310

VIII.5.10. USE 311

VIII.5.11. CØPY et INCLUDE 314

EXERCICES DU CHAPITRE VIII 315

BIBLIOTHEQUE DU CERIST

IX. LE TRI

IX.1. GENERALITES 323

IX.2. DESCRIPTION DU FICHIER DE TRI DANS LA DATA DIVISION 324

IX.3. ORDRES DE LA PROCEDURE DIVISION POUR EFFECTUER UN TRI 325

IX.3.1. SØRT 325

IX.3.2. {DESCENDING / ASCENDING} KEY *nom-donnée-1* [*nom-donnée-2*] 326

IX.3.3. USING *nom-fichier-1* 326

IX.3.4. GIVING *nom-fichier-2* 327

IX.3.5. INPUT PRØCEDURE *nom-procedure-1* [THRU *nom-procedure-2*] . 327

IX.3.6. ØUTPUT PRØCEDURE *nom-procedure-3* [THRU *nom-procedure-4*] 328

X. LES SOUS-PROGRAMMES

X.1. CHEZ CONTROL DATA 334

X.1.1. Ecriture d'un sous-programme en COBOL 334

X.1.2. Appel d'un sous-programme COBOL 335

X.1.3. Appel d'un sous-programme écrit dans un langage autre que COBOL 336

X.2. CHEZ IBM 336

X.2.1. Ecriture d'un sous-programme COBOL 336

X.2.2. Appel d'un sous-programme 337

X.3. CHEZ UNIVAC 338

X.3.1. Ecriture d'un sous-programme COBOL 338

X.3.2. Appel d'un sous-programme COBOL 339

X.3.3. Appel d'un sous-programme écrit dans un langage autre que COBOL 339

BIBLIOTHEQUE DU CERIST

X.4. CHEZ HONEYWELL 340

X.4.1. Ecriture d'un sous-programme en COBOL 340

X.4.2. Appel d'un sous-programme 341

CORRIGES DES EXERCICES DU CHAPITRE IV 343

CORRIGES DES EXERCICES DU CHAPITRE VII 347

CORRIGES DES EXERCICES DU CHAPITRE VIII 358

PROBLEMES COBOL 365

INDEX 389

BIBLIOTHEQUE DU CERIST

AVERTISSEMENT

BIBLIOTHEQUE DU CERIST

NOTATIONS UTILISEES DANS CET OUVRAGE POUR LA PRESENTATION DES ORDRES COBOL

Dans chacune des descriptions des ordres Cobol, les symboles suivants sont utilisés :

[ ] Tous les mots ou les parties d'un ordre placé entre [ ] peuvent être inclus ou omis dans l'ordre Cobol au choix du programmeur.

{ } Dans la description d'un ordre Cobol, certaines informations sont placées les unes sous les autres entre { } . Ces informations représentent diverses options utilisables. Une seule de ces options doit être choisie, les autres devant être omises.

... Ces points de suspension lorsqu'ils figurent derrière des [ ] ou des { }, signifient que le contenu des [ ] ou des { } peut être répété un certain nombre de fois.

—— Tous les mots qui sont soulignés sont absolument obligatoires. Ils devront donc figurer dans l'ordre Cobol, à moins que la partie de l'ordre qui les contient soit facultative. Dans l'écriture de son programme, le programmeur ne doit rien souligner.

MAJUSCULES

Tous les mots écrits en majuscules sont des mots-clé. Lorsqu'ils sont soulignés ils sont obligatoires alors que lorsqu'ils ne le sont pas ils sont facultatifs.

italique

Toutes les inscriptions en italique correspondent à des informations que doit fournir le programmeur (exemple *nom-donnée*, *littéral non-numérique*, etc.

Caractères spéciaux

Bien que les caractères spéciaux (= + - *, etc.) ne soient pas soulignés, ils sont obligatoires dans l'écriture de certains ordres Cobol.

Ponctuation

Lorsque la ponctuation est mentionnée dans un ordre Cobol, cela signifie qu'elle est obligatoire et qu'elle fait partie de la structure de l'ordre.

BIBLIOTHEQUE DU CERIST

I | INTRODUCTION

I.1. LA NOTION DE CERVEAUX ELECTRONIQUES

Il suffit d'ouvrir n'importe quels journaux de la grande presse ou n'importe quelle revue scientifique ou technique pour lire souvent sous des titres fort percutants, les prouesses de ces monstres électroniques que très malencontreusement on appelle "cerveaux électroniques".

Ces machines électroniques numériques, ces "ordinateurs" si l'on veut les appeler par leur nom, sont absolument polyvalents et sont utilisés à l'heure actuelle dans tous les domaines scientifiques et techniques ainsi que dans nos activités sociales quotidiennes.

Cette très regrettable dénomination de "cerveaux électroniques" fait souvent croire aux lecteurs que ces machines sont douées d'une intelligence, voire d'une intelligence supérieure, qui leur permet de traiter des problèmes que l'homme ne saurait résoudre. Il est bien certain que, si ces ordinateurs sont capables de résoudre avec une rapidité foudroyante et avec une probabilité quasi nulle d'erreur des problèmes qui nécessiteraient plusieurs années à un homme exécutant les calculs à la main, elles restent des machines absolument dépourvues d'esprit de raison et d'intelligence.

Un ordinateur ne fait rien de sa propre initiative, il est absolument incapable de créer quoi que ce soit, il ne fait que ce qu'on lui a dit de faire et rien d'autre.

Pour illustrer ces dires, on peut citer ici une histoire déjà bien connue.

Il s'agit d'un riche industriel qui voulant se mettre à "l'heure de l'Informatique" avait décidé de faire installer chez lui un ordinateur. Dès que la machine fut mise en place et prête à fonctionner, pour juger de sa soi-disant intelligence, il lui fit poser le problème suivant :

- j'ai deux montres, l'une cassée par conséquent qui est arrêtée, l'autre retarde très régulièrement de dix minutes par jour, quelle est celle qui m'est la plus utile ?

BIBLIOTHEQUE DU CERIST

La réponse de l'ordinateur fut la suivante : la montre la plus utile est celle qui est arrêtée, car elle est à l'heure deux fois par jour, alors que celle qui retarde de dix minutes par jour n'est à l'heure qu'une fois tous les 72 jours.

Voilà le genre de déduction que l'on peut faire faire à un ordinateur. Docilement, il exécutera le processus absurde de comparaison qu'on lui a donné, car lorsqu'il travaille, il ne sait pas ce qu'il fait. Ce n'est qu'une machine qui exécute tout ce qu'on lui dit de faire sans le moindre discernement comme toute machine, comme par exemple, une automobile qui sans l'ombre d'une hésitation va s'écraser contre un arbre, si son conducteur la conduit sur cet arbre.

Le propre de l'homme est de posséder l'esprit, l'intelligence, le pouvoir de création et d'innovation. Par contre, une de ses grandes faiblesses,est sa très grande imperfection dans l'exécution des travaux routiniers. Lorsqu'il doit effectuer de longs calculs, à la main par exemple, il les exécute lentement, se fatigue et finit immanquablement par faire des erreurs.

L'ordinateur par son pouvoir d'exécution extrêmement rapide et sûr apporte à l'homme ce qui lui manque. C'est donc l'outil idéal qui lui permettra d'exprimer son talent de créateur, de novateur et d'inventeur.

L'homme pense, l'ordinateur exécute, leur association permet la réalisation de travaux gigantesques et est en train de transformer très notablement la vie sociale des individus.

Mais qu'est-ce donc que ces machines au pouvoir d'exécution si prestigieux, que sont donc exactement ces ordinateurs ?

I.2. LES ORDINATEURS ET LEURS LANGAGES

Nous allons définir les ordinateurs en les comparant à leurs ancêtres connus de tous : les machines à calculer de bureau.

Une machine à calculer de bureau est constituée de trois organes principaux, comme le montre la figure 1.

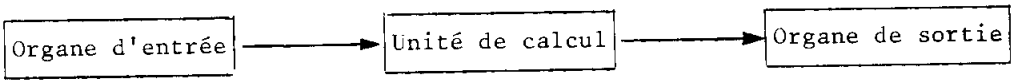

Fig. 1.

BIBLIOTHEQUE DU CERIST

L'organe d'entrée est constitué d'un clavier sur lequel l'utilisateur frappe les données de son problème. Cet organe d'entrée permet donc comme son nom l'indique d'entrer dans la machine les valeurs numériques sur lesquelles seront effectués les calculs.

L'unité de calcul permet de réaliser automatiquement les quatre opérations arithmétiques classiques : l'addition, la soustraction, la multiplication et la division. Le fait que la division se fasse automatiquement dans les machines à calculer modernes implique, qu'elles soient capables de faire des comparaisons puisque le calcul d'une division se fait par une succession de soustractions et de comparaisons entraînant les décalages nécessaires pour la poursuite du calcul de la division.

L'unité de calcul d'une machine à calculer de bureau permet donc d'effectuer les opérations arithmétiques classiques et des opérations logiques.

L'organe de sortie est constitué soit d'une petite machine à écrire pour les machines à calculer dites imprimantes, soit d'un registre où apparaissent les résultats des opérations.

Un ordinateur est en premier lieu une machine à calculer classique. En effet, il possède :

- Un ou plusieurs organes d'entrée : clavier, lecteur de cartes ou de bandes perforées, lecteur de bandes magnétiques, etc.

Ces organes d'entrée ont exactement la même mission que le clavier de la machine à calculer de bureau. Ils permettent seulement d'entrer dans l'ordinateur les informations qui doivent être traitées pour l'élaboration du problème que l'on a à résoudre. Le clavier qui permettait de fournir les données aux tout premiers ordinateurs a disparu, parce que trop lent, au profit des lecteurs de cartes et de bandes de beaucoup plus rapides.

- Une unité de calcul qui, tout comme celle de la machine à calculer de bureau, permet d'effectuer les quatre opérations arithmétiques, des comparaisons permettant de faire des opérations logiques et booléennes.

En ce qui concerne cette unité de calcul, il n'y a là non plus aucune différence fondamentale entre celle d'une machine à calculer de bureau et celle d'un ordinateur.

- Un ou plusieurs organes de sortie : machine à écrire, imprimante, perforateur de cartes ou de bandes perforées, dérouleur de bandes magnétiques, etc.

BIBLIOTHEQUE DU CERIST

Ces organes de sortie ont exactement le même rôle que la machine à écrire de la machine à calculer de bureau. Ils ont pour mission de restituer à l'utilisateur de l'ordinateur les résultats de son problème.

Certes, au fur et à mesure des progrès de l'électronique, les ordinateurs sont équipés d'unités de calcul de plus en plus rapides. Il est aussi nécessaire que les liaisons"milieux extérieurs-ordinateurs" soient réalisées par des organes d'entrée-sortie beaucoup plus puissants que ceux d'une machine à calculer de bureau.

Mais il faut bien constater que, d'après ce que nous venons de dire, il n'y a jusqu'ici aucune différence fondamentale entre un ordinateur et une machine à calculer de bureau.

Or, nous avons dit précédemment que ces ordinateurs extrêmement polyvalents étaient en train de provoquer une véritable révolution, chose qui ne s'est jamais produite au préalable avec les machines à calculer de bureau qui existent pourtant depuis Pascal, c'est-à-dire, depuis plus de trois siècles. Cela veut dire que les ordinateurs possèdent quelque chose qui leur est bien spécifique et qui les caractérise et les différencie de leurs ancêtres.

Qu'est-ce donc que cette chose ? C'est la "Mémoire".

Certes, il n'est pas de notre propos de faire un exposé sur la technologie de la Mémoire d'un ordinateur, nous renvoyons le lecteur intéressé par cette technologie à des ouvrages spécialisés. Nous nous contenterons de dire que la Mémoire d'un ordinateur contient un certain nombre de "cases" qui permettent de stocker sous forme codée toutes les informations qui doivent être traitées au cours de l'exécution du problème.

Chacune de ces cases est constituée d'un ensemble de petits tores magnétiques situés aux noeuds d'un maillage de fils qui permettent en fonction des courants qui les traversent de donner aux tores deux orientations d'aimantation possibles.L'une représente conventionnellement la valeur zéro, l'autre la valeur 1; aussi les ordinateurs travaillent-ils dans le système binaire. Chacun de ces tores qui prend la valeur 0 ou 1 s'appelle "position binaire" ou "bit" (abréviation de binary digit).

La Mémoire d'un ordinateur va donc permettre :

1) le stockage des informations nécessaires pour mener à bien le problème que l'on doit résoudre;

2) le stockage des résultats intermédiaires et des résultats définitifs;

3) le stockage du programme.

BIBLIOTHEQUE DU CERIST

Qu'est-ce donc que le programme ?

Le programme est la suite logique et ordonnée de toutes les opérations,au sens large du terme,que devra exécuter la machine pour résoudre le problème.

Aussi, pour qu'un problème puisse être donné à un ordinateur, il est absolument indispensable qu'il lui soit présenté sous forme d'un programme, c'est-à-dire sous la forme d'une suite séquentielle d'opérations arithmétiques, logiques et également d'opérations d'échanges d'informations entre le milieu extérieur et l'ordinateur.

Ce n'est que lorsqu'un problème est mis sous cette forme qu'il peut être résolu par un ordinateur. Certes la décomposition d'un problème en cette suite logique d'opérations présente des difficultés. Pour faciliter ce travail, le programmeur,avant d'écrire son programme en fera l'organisation sous forme schématisée:c'est ce que l'on appelle l'établissement de l'organigramme (voir chapitre III.).

Une fois le programme écrit, il est enregistré dans la Mémoire de l'ordinateur sous forme codée binaire. Chaque opération que doit exécuter la machine est donc représentée dans le programme par ce que l'on appelle une instruction. La suite séquentielle de ces instructions constitue le programme. Associé à la Mémoire de l'ordinateur, un organe appelé "organe de commande" permet sans la moindre intervention humaine, d'exécuter cette suite d'instructions et ainsi de résoudre le problème posé. La configuration très schématique d'un ordinateur peut donc se représenter par la figure 2.

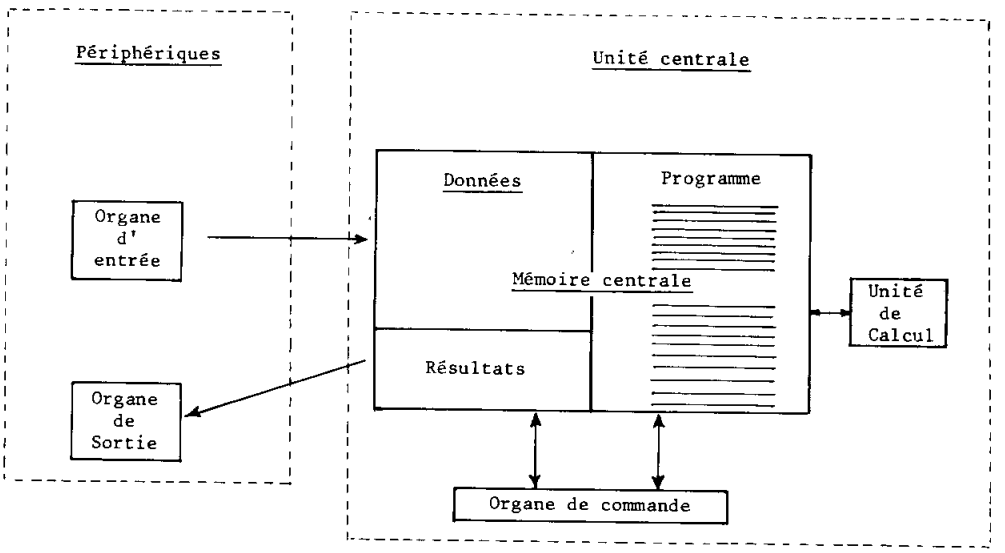

Fig. 2.

BIBLIOTHEQUE DU CERIST

Dans la Mémoire centrale d'un ordinateur, tout (le programme, les données et les résultats) est donc représenté sous forme codée binaire.

Une question se pose, immédiatement, doit-on écrire le programme sous forme codée binaire ? Heureusement, non. Toutefois, il faut quand même écrire le programme dans un langage interprétable par la machine.

Il existe deux grandes catégories de langages :

- les langages autocodes encore appelés langage d'assembleur;

- les langages évolués.

Les langages autocodes qui étaient les seuls à exister au début de l'ère des ordinateurs, présentent l'avantage d'être spécifiques de chaque machine. Ainsi, ils tiennent compte de la technologie propre de chaque ordinateur et permettent de réaliser les meilleures performances. Du fait même de leur spécificité, ils ont l'énorme inconvénient d'être différents d'une machine à l'autre. De telle sorte que des programmes écrits en langage autocode d'un certain ordinateur ne peuvent pas être exécutés sur un autre. Aussi, lorsque l'on change le matériel, est-on obligé de réécrire entièrement le programme.

Le programme écrit sur des feuilles conçues à cet effet et appelées "bordereaux de perforation", est perforé sur des cartes (chaque caractère étant représenté par un certain codage de perforation, voir chapitre II).

Nous avons dit précédemment que le programme qui est enregistré en mémoire centrale de la machine et qui doit être exécuté, doit se présenter sous forme codée binaire. Or le programmeur écrit le programme en langage d'assembleur, qui n'est absolument pas le langage codé binaire interprétable par la machine. Il est donc absolument indispensable que l'ordinateur, automatiquement, c'est-à-dire sans aucune intervention de la part du programmeur, puisse traduire ce programme du langage autocode en son propre langage machine codé binaire. Ceci se fait à l'aide d'un programme qui fait partie du système de la machine, appelé "assembleur".

Ainsi, à partir du paquet de cartes "programme source" l'assembleur va générer un programme en langage binaire. A l'aide d'un autre programme du système appelé "chargeur" ce programme binaire sera implanté en mémoire centrale de la machine et sera prêt à être exécuté.

BIBLIOTHEQUE DU CERIST

La traduction du programme source en langage machine est représentée par la figure 3.

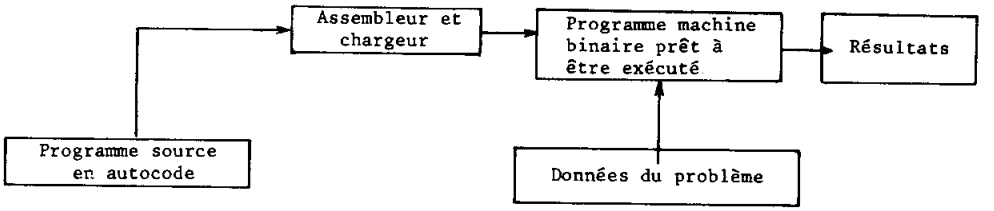

Fig. 3.

Pour écrire un programme en langage autocode, le programmeur doit descendre au niveau de chaque opération élémentaire et écrire tous les ordres que doit exécuter la machine au moyen de codes que le constructeur a mis à sa disposition. Malgré un choix judicieux de codes qui sont généralement les mnémoniques de ce qu'ils doivent faire exécuter à la machine, l'utilisation des langages autocodes est relativement délicate et lourde, car de tels langages ne sont pas "très parlants".

Toutes ces raisons alliées au fait de la spécificité des langages autocodes ont fait que des langages évolués, non spécifiques d'une installation, mais au contraire utilisables sur tous les ordinateurs, ont été mis au point.

De tels langages sont d'une utilisation extrêmement simple parce qu'ils sont très proches du langage couramment employé par tous. Ces langages permettent à n'importe quel individu doté d'un esprit logique normal d'écrire un programme et aussi d'utiliser, comme outil de travail, n'importe quel ordinateur.

Les quatre langages évolués les plus connus à l'heure actuelle sont :

Le FORTRAN : FORmula TRANslator

L'ALGOL : ALGOrithmic Language

Ces deux langages ont été créés pour le calcul scientifique et se rapprochent beaucoup de la formulation mathématique classique.

Le COBOL : COmmon Business Oriented Language

qui a été spécialement créé pour résoudre les problèmes de gestion. Il se rapproche énormément du langage parlé courant. C'est ce langage qui fait l'objet de cet ouvrage.

BIBLIOTHEQUE DU CERIST

Le PL 1 : Programming Language n°1

qui a été créé pour pouvoir traiter à la fois des problèmes scientifiques et des problèmes de gestion. Le PL 1 est donc un langage que l'on peut qualifier d'universel; toutefois, il faut souligner qu'il n'est pas encore opérationnel sur tous les ordinateurs.

L'ordinateur étant capable d'exécuter les instructions uniquement lorsqu'elles sont mises sous forme du langage machine codé binaire, il faut bien entendu, comme pour les langages autocodes, que les machines disposent d'un traducteur qui, à partir du programme source écrit dans un des langages évolués, traduit ce programme en un programme en langage machine codé binaire. Cette traduction se fait à l'aide d'un programme du système de l'ordinateur appelé "Compilateur".

Généralement,la traduction d'un programme source écrit en langage évolué,en un programme codé binaire (programme objet) se fait comme le représente la figure 4.

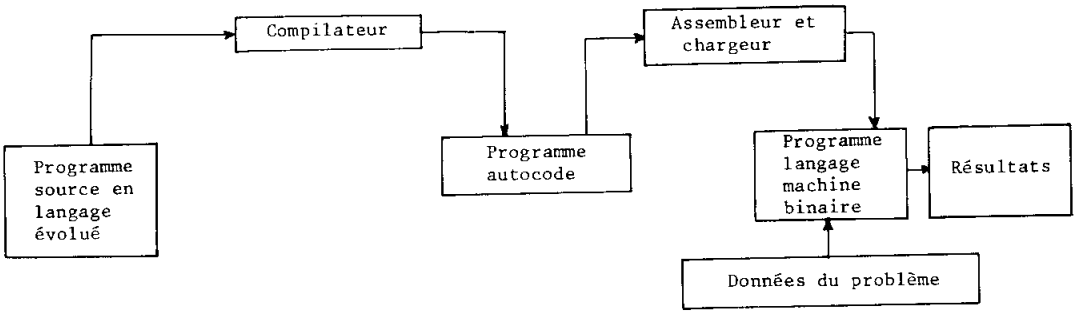

Fig. 4.

Remarque.
Certains compilateurs génèrent directement à partir du programme source un programme en langage machine binaire, sans générer un programme intermédiaire en langage autocode.

Nous voyons qu'un programme écrit en langage évolué nécessite généralement une étape de plus pour sa traduction que lorsque le programme est écrit en langage autocode.

Nous pouvons dire en conclusion que chaque type de langage a ses avantages et ses inconvénients.

BIBLIOTHEQUE DU CERIST

Le programmeur utilisant un langage autocode pourra écrire un programme très astucieux et beaucoup plus rapide à l'exécution que s'il l'écrivait dans un langage évolué. Mais l'élaboration de ce programme en langage autocode lui demandera beaucoup plus d'efforts et de temps que s'il le rédigeait en langage évolué.

Le gros avantage des langages évolués est que, d'une part, ils sont utilisables sur tous les ordinateurs et que d'autre part, leur syntaxe est extrêmement simple, puisque se rapprochant du langage courant.

C'est la raison pour laquelle, ces langages évolués sont très utilisés dans la pratique, en particulier le COBOL, pour résoudre tous les problèmes de gestion.

BIBLIOTHEQUE DU CERIST