LUCA-DAN ȘERBĂNAȚI
VALENTIN CRISTEA
FLORICA MOLDOVEANU
VALERIU IORGA

Programarea sistematică în limbajele PASCAL și FORTRAN

Systematic programming in PASCAL and FORTRAN

Computer programming is a complex and difficult task. In order to reduce the programming effort, optimisation methods are required.

Among the techniques used in the last decade, structured programming and the developement of programs by stepwise refinement are the most widely accepted.

This work presents the basic ideas concerning the methods and techniques above mentioned, using two popular languages: PASCAL and FORTRAN. The process of algorithm development is quite similar for these two languages, despite the major differences between them.

A lot of simple, intuitive examples are given. Projects and programs for some complex problems are contained in the last chapter. A great number of problems are submitted to the reader by each chapter.

The work is intended to the use of students, engineers, programmers, workers in the field of computer programming and also to everyone who wants to learn about the two languages.

CONTENTS

1. Introduction

2. Program design

3. Programming in PASCAL

4. Programming in FØRTRAN

5. Program debugging

6. Appendix

CUVÎNT ÎNAINTE

Volumul de față apare într-un moment în care activitatea de elaborare de programe informatice în țara noastră tinde să devină o producție cu caracter industrial. Autorii au redactat o lucrare cu deosebite calități didactice pentru limbajele PASCAL și FORTRAN avînd în vedere cerințele unei înalte profesionalități în activitatea de programare. Ei arată, în mod just, în primul capitol: „multă vreme programarea a fost practicată ca un meșteșug ... Învățarea programării a însemnat mult timp doar obținerea unei practici de programare și nimic altceva.

Acest stil de programare nu mai corespunde cerințelor actuale datorită atît complexității și diversității problemelor de rezolvat cu calculatorul, cît și necesității de creștere a eficienței activității de programare sub aspectul timpului de elaborare a programelor și al calității lor. Deși programarea continuă să solicite intuiția și experiența programatorului, ea tinde să devină o activitate sistematică, cu etape bine precizate, cu metode și instrumente de lucru adecvate fiecărei etape".

Lucrarea este scrisă avîndu-se în vedere un asemenea mod de abordare, specific cerințelor formulate pentru o industrie de programe[1]*. Autorii insistă, dintre toate etapele scrierii propriu-zise a programelor, asupra proiectării programelor și depanării programelor, adică asupra fazelor care preced și succed scrierii (codificării) programelor. Se insistă, deci, asupra elaborării unor programe corecte. Ce înseamnă însă un program corect ?*

După Conway[2] *(1978) există opt înțelesuri ale unui program corect.*

I. Nu conține erori sintactice.

II. Nu conține erori de compilare sau nu se manifestă nereușita rulării unui program.

III. Există date de test pentru care programul dă răspunsuri corecte.

IV. Pentru categorii tipice de date de test programul dă răspunsuri corecte.

V. Pentru categorii dificile de date de test programul dă răspunsuri corecte.

---

[1] V. comunicările sesiunii științifice, Viitorul industriei de programe. Academia Republicii Socialiste România, București, 22 aprilie, 1983.

[2] Conway, R. *A primer on disciplined programming.* Winthrop Publishers, Cambridge, Mass., 1978 (citat după M. V. Zelkowitz, Shaw, A. C., Gannon, J. D. *Principles of software engineering and design.* Englewood Cliffs, New Jersey, Prentice-Hall, 1979, p. 27–28).

VI. Pentru toate categoriile posibile de date, conform specificării problemei, programul dă răspunsuri corecte.

VII. În condițiile datelor de la pct. VI, dar și pentru toate condițiile probabile ale unor intrări eronate programul dă răspunsuri corecte.

VIII. Pentru orice date la intrare, posibile, programul dă răspunsuri corecte.

Cele opt înțelesuri de mai sus sînt de fapt nivele de corectitudine. Atingerea nivelului V reprezintă o cerință minimă pentru un program de calitate superioară iar obținerea nivelului VI poate fi considerată mai mult decît mulțumitoare., Nivelele VII și VIII sînt mai pretențioase, implicînd o anumită înțelegere eventual o anumită inteligență a programului pentru a detecta unele erori ale datelor de intrare.

După ce programul este livrat sau pus în funcție începe o nouă etapă a ciclului lui de viață și anume întreținerea *(mentenanța) programului. Întreținerea unui program a devenit, în cazul unei industrii de programe, o activitate recunoscută, planificată, căreia i se dedică echipe de specialiști. Programul trebuie să fie proiectat și elaborat în așa fel încît întreținerea să fie cît mai ușoară și mai puțin costisitoare. Și cu toate acestea există o mare deosebire între costul întreținerii unui program informatic și al unui echipament (fig.1.1)*[1]*. Întreținerea unui program înseamnă modificarea lui pentru a „repara" unele neajunsuri a-i modifica comportamentul.*

Un program informatic este un obiect complex, uneori foarte complex. Fiecare instrucțiune care execută o anumită operațiune este echivalentul unei piese dintr-un ansamblu mecanic sau al unui subansamblu. Un program cu 5 000 de instrucțiuni este echivalentul unui sistem mecanic din 5 000 de piese!

Complexitatea multor programe de calcul face din ele obiecte de studiu de mare interes științific. Obiectul program informatic se dovedește o problemă nouă de realitate cu care omul și societatea se confruntă astăzi. Studiul unor asemenea obiecte prin metode științifice adecvate se găsește abia la început.

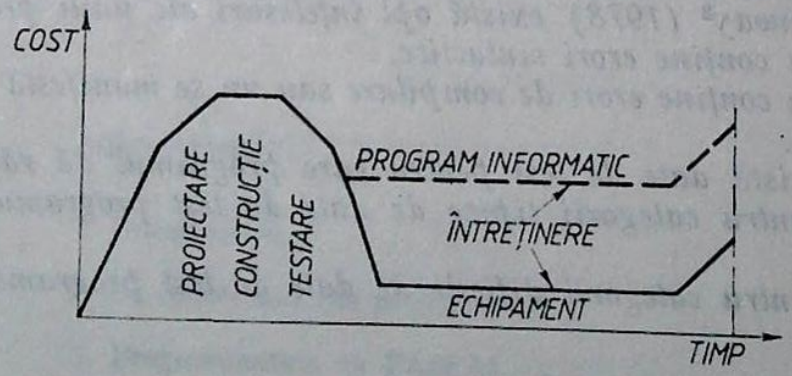

Fig. 1.1. Costul echipamentelor și programelor informatice în diferite faze ale ciclului de viață.

[1] Allworth, S. T. *Introduction to real-time software design.* London, Macmillan Press, 1981.

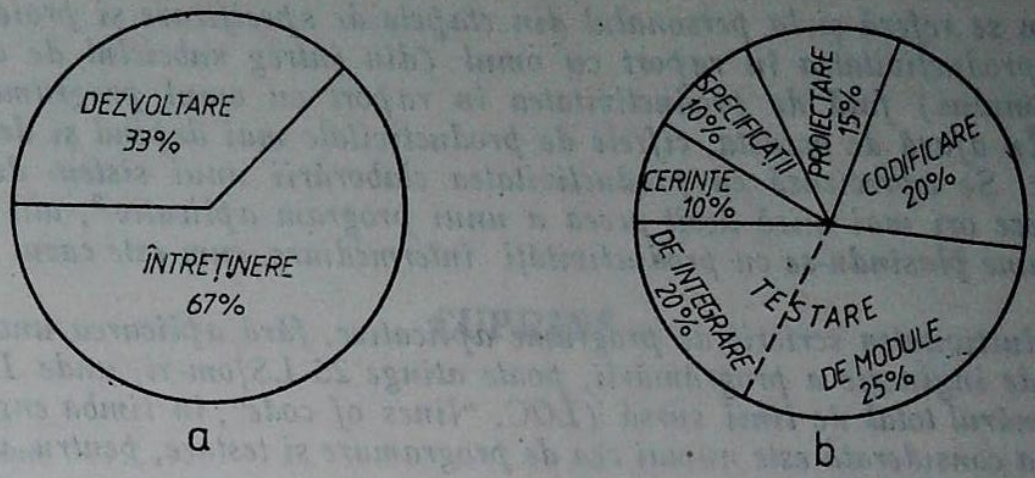

Fig. 1.2. Eforturile de întreținere și dezvoltare în ciclul de viață al unui program

Programele informatice interesează, în actuala etapă, în primul rînd din punctul de vedere al ingineriei programării[1], *un domeniu ceva mai larg decît scrierea de programe și chiar decît cele ale specificării, proiectării și întreținerii programelor. Ingineria programării se referă și la managementul activității de programare, la metodele inginerești de organizare și la aspectele economice ale programelor. De aceea abordarea programării într-adevăr în cadrul unei viziuni de* activitate sistematică *se impune cu stringență.*

Din punct de vedere economic, în ciclul de viață al unui program complex. efortul de întreținere reprezintă circa 67%, restul de 33% reprezentînd efortul de dezvoltare (fig. 2 a). Efortul de dezvoltare este, în general, distribuit după cum se arată în fig. 2 b[2]. *Se poate observa cum efortul de testare reprezintă circa 45% din efortul total de dezvoltare, în timp ce proiectării (scrierea algoritmilor) îi revine 15% iar codificării 20%. Fazele inițiale de scriere a cerințelor (de fapt adaptarea lor prin discuții cu beneficiarul) și a specificațiilor reprezintă 20% din efort.*

Consumul de forță de muncă pe care societatea îl depune în elaborarea programelor ridică și problema productivității muncii *în acest domeniu. În ultimul timp se discută problema definirii unor indicatori pentru aprecierea productivității muncii de elaborare de programe. Un indicator care se dovedește deosebit de util*[3] *este acela al* numărului de linii sursă/om *și zi, unde prin linie sursă se înțelege fie o linie-instrucțiune fie o linie-comentariu. Prin* om *se înțelege uneori numai programatorul care scrie liniile sursă și care testează programul,*

---

[1] Văduva, I., Baltac, V. ș.a. *Ingineria programării.* Studii și cercetări de calcul economic și cibernetică economică, 1982, nr. 4 și nr. 5.

Baltac, V. *Considerații privind o industrie de programe pentru calculatoare electronice,* Revista economică, 1979, nr. 5.

[2] Baltac, V. *Tehnica electronică de calcul și dezvoltarea României.* în volumul *Noile tehnologii de vîrf și societatea.* București, Editura Politică, Colecția „Idei contemporane", 1983, p. 51—75.

[3] Zelkowitz, M. V., Shaw, A. C., Gannon, J. D., *Principles of software engineering and design.* Prentice Hall, Englewood Cliffs, New Jersey, 1979, p. 2—3.

5. *Probleme propuse spre rezolvare* 102

Partea a treia 106

Programarea în PASCAL 106

1. *Introducere* 107
2. *Structura generală a programelor PASCAL* 111
3. *Tipuri de date (I). Tipuri simple* 111
   3.1. Clasificarea tipurilor de date 112
   3.2. Tipuri scalare (tipuri enumerate) 114
   3.3. Tipuri scalare predefinite (standard) 114
      3.3.1. Tipul întreg 116
      3.3.2. Tipul real 116
      3.3.3. Tipul logic 117
      3.3.4. Tipul caracter 118
   3.4. Tipuri subdomeniu 119
4. *Declarații în PASCAL* 119
   4.1. Clasificarea declarațiilor 120
   4.2. Declarația etichetelor 120
   4.3. Definiția constantelor 121
   4.4. Definiția tipurilor 122
   4.5. Declarația variabilelor 123
   4.6. Declarația subprogramelor 123
5. *Instrucțiuni în PASCAL* 123
   5.1. Clasificarea instrucțiunilor 124
   5.2. Instrucțiunea compusă 124
   5.3. Instrucțiunea de atribuire 127
   5.4. Citirea și scrierea datelor 128
      5.4.1. Citirea datelor din fișierul INPUT (procedurile READ și READLN) 129
      5.4.2. Scrierea datelor în fișierul OUTPUT (procedurile WRITE și WRITELN) 131
   5.5. Instrucțiuni condiționale 131
      5.5.1. Instrucțiunea IF 133
      5.5.2. Instrucțiunea CASE 134
   5.6. Instrucțiunea GOTO 136
   5.7. Instrucțiunile repetitive 136
      5.7.1. Instrucțiunea WHILE 137
      5.7.2. Instrucțiunea REPEAT 140
      5.7.3. Instrucțiunea FOR 141
   5.8. Instrucțiunea vidă 141
   5.9. Considerații privind scrierea programelor 145
6. *Tipuri de date (II). Tipuri structurate* 145
   6.1. Structura de tablou 145
      6.1.1. Tipuri și variabile tablou 150
      6.1.2. Structuri „împachetate" 152
   6.2. Structura de înregistrare 152
      6.2.1. Tipuri și variabile-înregistrare 156
      6.2.2. Instrucțiunea WITH 157
      6.2.3. Înregistrări cu variante 161
   6.3. Tipuri și variabile mulțime 165
7. *Subprograme* 165
   7.1. Proceduri PASCAL 167
   7.2. Parametri formali și parametri efectivi 173
   7.3. Domeniul de valabilitate al variabilelor 176
   7.4. Funcții PASCAL 181
8. *Tipuri de date în PASCAL (III)* 181
   8.1. Tipuri referință 181
      8.1.1. Tipuri referință și variabile-indicator 181