| Fişa suspiciunii de plagiat / Sheet of plagiarism's suspicion | Indexat la: 0136/00 |
|---|---|

| Opera suspicionată (OS)<br>Suspicious work | Opera autentică (OA)<br>Authentic work |
|---|---|

| | |
|---|---|
| OS | Lacrămă, L.Dan. *Limbajul C. Notițe de curs.* Timişoara: Centrul de multiplicare, Universitatea Politehnica. 2004. |
| OA | Cristea, V., Giumale, C., Kalisz, E., Pănoiu, A. *Limbajul C Standard.* Bucureşti: Teora. 1992. |

| Incidența minimă a suspiciunii / Minimum incidence of suspicion | |
|---|---|
| p.09:01s- p.111:20d | p.09:01s-p.111:20d |
| Fişa întocmită pentru includerea suspiciunii în Indexul Operelor Plagiate în România de la Sheet drawn up for including the suspicion in the Index of Plagiarized Works in Romania at www.plagiate.ro | |

**Notă**: p.72:01s-p.87:03d semnifică textul de la primul rând al coloanei din stânga de la pag.72 până la rândul al treilea al coloanei din dreapta de la pag.87.

# Calculatoare personale

Valentin Cristea
Cristian Giumale
Eugenia Kalisz
Alexandru Pănoiu

Teora

Valentin Cristea
Cristian Giumale

Eugenia Kalisz
Alexandru Pănoiu

# Limbajul C standard

Editura TEORA
1992

# 1. Trecere în revistă a limbajului C

## 1.1. Generalităţi

Un limbaj de programare acţionează ca o interfaţă între universul real al problemei de rezolvat şi programul de rezolvare, punînd la dispoziţie o serie de elemente constructive (**entităţi**) şi legi de combinare a acestora, prin care elementele problemei şi acţiunile de rezolvare pot fi reprezentate şi prelucrate la nivelul programului. A construi un program de rezolvare a problemei înseamnă, în esenţă, a găsi o modalitate de agregare a acestor **entităţi**, în aşa fel încît rezultatul - anume programul - atunci cînd este executat de calculator să constituie o replică a procesului pe care un rezolvitor uman l-ar executa pentru a rezolva problema.

Limbajul de programare, privit ca interfaţă între problema de rezolvat şi programul de rezolvare, nu asigură, în marea majoritate a cazurilor, o trecere directă de la specificarea problemei la program, ci impune considerarea şi, datorită interdependenţelor, reconsiderarea mai multor etape.

În primul rînd, există o etapă de analiză şi abstractizare a problemei, de identificare a **obiectelor** implicate în rezolvare şi a acţiunilor de transformare corespunzătoare acestora. Un **obiect** este considerat aici ca element esenţial al unei probleme, element ce poate fi analizat independent, prelucrabil prin operatori specifici, în funcţie de tipul său, şi a cărui stare curentă influenţează starea problemei din punctul de vedere al rezolvării acesteia. Rezultatul acestei prime etape este un aşa numit **univers abstract al problemei (UP)**, care pune în evidenţă mulţimea tipurilor de obiecte, a obiectelor, a relaţiilor între obiecte şi a restricţiilor de prelucrare necesare rezolvării problemei.

Urmează apoi etapele găsirii unei metode de rezolvare acceptabile, precizării exacte a operatorilor de prelucrare proprii obiectelor din UP, elaborării algoritmului de rezolvare şi, în final, codificării algoritmului şi reprezentării obiectelor din UP folosind **entităţile** şi legile lor de combinare oferite de limbaj.

Aceste etape, chiar şi cele iniţiale, sînt însă influenţate într-o măsură, deseori importantă, de natura limbajului de programare, de **entităţile** prin care un program poate fi construit folosind limbajul, deşi se urmăreşte ca influenţa să fie cît mai mult minimizată. Orice limbaj este menit să faciliteze rezolvarea unui anumit gen de probleme, se pretează mai bine unui anumit gen de algoritmi. În particular, C-ul a fost dezvoltat mai ales pentru elaborarea programelor de sistem, care au, în cea mai mare parte, caracter nenumeric, importante fiind aici:

a) uşurinţa de reprezentare a obiectelor cu caracter nenumeric şi a relaţiilor dintre acestea;

b) capacitatea de a prelucra obiecte dinamice din punctul de vedere al duratei lor de viaţă, proprietăţilor şi inter-relaţiilor stabilite în cursul prelucrării.

c) capacitatea de a exploata caracteristicile maşinii de calcul utilizate în scopul controlului strict al performanţelor programului relativ la raportul memorie consumată / viteză de execuţie, deci de a putea programa atît la nivel înalt, cît şi la un nivel apropiat codului maşină;

d) asigurarea unei interfeţe cît mai transparente cu sistemul de operare al maşinii utilizate.

Din motivele arătate, credem că este bine, la început, să prezentăm ceea ce este remarcabil în C şi, totodată, ceea ce este comun şi altor limbaje de programare din aceeaşi clasă cu C-ul, uneori chiar foarte diferite. De asemenea, succinta "vizitare" a limbajului va aduce la suprafaţă concepte fundamentale în programare: constantă, variabilă, tip de dată, valoare dreaptă si valoare stîngă a unei variabile, domeniu de valabilitate al numelor, durată de viaţă a variabilelor, spaţii ale numelor etc., concepte care trebuie stăpînite, pentru o deplină înţelegere şi o utilizare corectă a limbajului. Aceste concepte sînt repertoriate concentrat în anexa A. Totodată, sperăm ca cititorul să fie capabil să scrie programe C simple chiar înainte de a parcurge prezentarea detaliată a limbajului.

Cei care nu au încă o experienţă deosebită de programare vor putea observa aici la ce trebuie să se aştepte în C; cei experimentaţi vor folosi, probabil, restul cărţii ca referinţă.

## 1.2. Structura unui program C. Modularizare

Spre deosebire de majoritatea limbajelor de programare convenţionale de nivel înalt, în C, un program nu conţine un "program principal". Cărămida de bază este aici funcţia, un program fiind construit dintr-o mulţime de funcţii grupate în **module program**.

Această caracteristică apropie într-o oarecare măsură C-ul de limbajele mai "neortodoxe" gen LISP şi SCHEME care, la rîndul lor, doresc - cu succes limitat - a fi limbaje funcţionale, anume, limbaje în care principalul element de construcţie a programelor este funcţia, considerată ca valoare obişnuită, prelucrabilă arbitrar într-un program, iar principalul mecanism de control este aplicarea funcţiilor asupra unor valori valide în limbaj. Bunăoară, într-un limbaj funcţional, compunerea a două funcţii pentru a obţine o nouă funcţie este o operaţie obişnuită. Am insistat puţin asupra acestui aspect pentru a arăta că C-ul nu este un limbaj funcţional în adevăratul sens al cuvîntului, chiar dacă