

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

Opera suspicionată (OS) Suspicious work	Opera autentică (OA) Authentic work
--	--

OS	Lacrămă, L.Dan. <i>Limbajul C. Notițe de curs</i> . Timișoara: Centrul de multiplicare, Universitatea Politehnica. 2004.
OA	Cristea, V., Giumale, C., Kalisz, E., Pănoiu, A. <i>Limbajul C Standard</i> . București: Teora. 1992.

Incidența minimă a suspiciunii / Minimum incidence of suspicion	
--	--

p.09:01s- p.111:20d	p.09:01s-p.111:20d
---------------------	--------------------

Fișa 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.

Universitatea "Politehnica" din Timișoara

Limbaajul C

- Notițe de curs -

conf. dr. ing. Lacrămă L. Dan

Timișoara 2001

1. Trecere în revistă a limbajului C

1.1. Generalități

Un limbaj de programare acționează ca o interfață între universul real al problemei de rezolvat și programul de rezolvare, punând la dispoziție o serie de elemente constructive (entități) și legi de combinare a acestora, prin care elementele problemei și acțiunile de rezolvare pot fi reprezentate și prelucrate la nivelul programului. A construi un program de rezolvare a problemei înseamnă, în esență, a găsi o modalitate de agregare a acestor entități, în așa fel încât rezultatul - anume programul - atunci când 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 interfață între problema de rezolvat și programul de rezolvare, nu asigură, în marea majoritate a cazurilor, o trecere directă de la specificarea problemei la program, ci impune considerarea și, datorită interdependențelor, reconsiderarea mai multor etape.

În primul rând, există o etapă de analiză și abstractizare a problemei, de identificare a obiectelor implicate în rezolvare și a acțiunilor de transformare corespunzătoare acestora. Un obiect este considerat aici ca element esențial al unei probleme, element ce poate fi analizat independent, prelucrabil prin operatori specifici, în funcție de tipul său, și a cărui stare curentă influențează starea problemei din punctul de vedere al rezolvării acesteia. Rezultatul acestei prime etape este un așa numit univers abstract al problemei (UP), care pune în evidență mulțimea tipurilor de obiecte, a obiectelor, a relațiilor între obiecte și a restricțiilor 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 și, în final, codificării algoritmului și reprezentării obiectelor din UP folosind entitățile și legile lor de combinare oferite de limbaj.

Aceste etape, chiar și cele inițiale, sînt însă influențate într-o măsură, deseori importantă, de natura limbajului de programare, de entitățile prin care un program poate fi construit folosind limbajul, deși se urmărește ca influența să fie cât 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) ușurința de reprezentare a obiectelor cu caracter nenumeric și a relațiilor dintre acestea;

b) capacitatea de a prelucra obiecte dinamice din punctul de vedere al duratei lor de viață, proprietăților și inter-relațiilor stabilite în cursul prelucrării.

c) capacitatea de a exploata caracteristicile mașinii de calcul utilizate în scopul controlului strict al performanțelor programului relativ la raportul memorie consumată / viteză de execuție, deci de a putea programa atât la nivel înalt, cât și la un nivel apropiat codului mașină;

d) asigurarea unei interfețe cât mai transparente cu sistemul de operare al mașinii utilizate.

Din motivele arătate, credem că este bine, la început, să prezentăm ceea ce este remarcabil în C și, totodată, ceea ce este comun și altor limbaje de programare din aceeași clasă cu C-ul, uneori chiar foarte diferite. De asemenea, succinta "vizitare" a limbajului va aduce la suprafață concepte fundamentale în programare: constantă, variabilă, tip de dată, valoare dreaptă și valoare stîngă a unei variabile, domeniu de valabilitate al numelor, durată de viață a variabilelor, spații ale numelor etc., concepte care trebuie stăpinite, pentru o deplină înțelegere și o utilizare corectă a limbajului. Aceste concepte sînt repertoriile concentrate 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 experiență deosebită de programare vor putea observa aici la ce trebuie să se aștepte în C; cei experimentați vor folosi, probabil, restul cărții ca referință.

1.2. Structura unui program C. Modularizare

Spre deosebire de majoritatea limbajelor de programare convenționale de nivel înalt, în C, un program nu conține un "program principal". Cărămida de bază este aici funcția, un program fiind construit dintr-o mulțime de funcții grupate în module program.

Această caracteristică apropie într-o oarecare măsură C-ul de limbajele mai "neortodoxe" gen LISP și SCHEME care, la rîndul lor, doresc - cu succes limitat - a fi limbaje funcționale, anume, limbaje în care principalul element de construcție a programelor este funcția, considerată ca valoare obișnuită, prelucrabilă arbitrar într-un program, iar principalul mecanism de control este aplicarea funcțiilor asupra unor valori valide în limbaj. Bunăoară, într-un limbaj funcțional, compunerea a două funcții pentru a obține o nouă funcție este o operație obișnuită. Am insistat puțin asupra acestui aspect pentru a arăta că C-ul nu este un limbaj funcțional în adevăratul sens al cuvîntului, chiar dacă