METALIMBAJUL ÎN SISTEMUL LIMBAJELOR DE PROGRAMARE

Olimpia VARGA
Universitatea „Ovidius" din Constanţa
olimpia_varga@yahoo.com

Abstract**:** In this paper we try to clarify the mechanisms of programming languages starting from the general concepts, the alphabet and vocabulary, syntax and the semantics of artificial languages compared to natural languages. Within programming languages we analyze metalanguage elements, the combinations of letters, numbers, word categories, the set of symbols and the commands used in the program.

Keywords: programming language, semantics, syntax, symbol, metalanguage

Un specialist în informatică sau un filolog specializat în lingvistică computaţională are imaginea mecanismelor de funcţionare a limbajelor de programare, după modelul limbilor naturale și după cum le este foarte familiar şi metalimbajul aferent.

Pentru nespecialişti, cele două tipuri de sisteme[1] – natural şi artificial –, precum și terminologia „gramaticii" după care se organizează fiecare nu își dezvăluie specificul şi modelele de autoreglare chiar atât de uşor. Ceea ce obturează suplimentar imaginea reluată a limbajelor, de la natural la artificial, este accepţia oarecum particulară a terminologiei de specialitate.

Dacă termeni precum „semantică", „sintaxă" etc. sunt relativ unitari în interpretarea ambelor tipuri de sisteme, există alţi termeni, precum „dicţionar", „bibliotecă", „limbaje de interogare"[2] etc. cu înţelesuri speciale în limbajele generate de inteligenţa artificială. În plus, aceasta din urmă își dezvoltă limbajele într-un ritm extrem de rapid, procesul de simplificare a procedurilor de analiză fiind invers proporţional cu complexitatea operaţiilor logice şi comunicaţionale pe care sunt capabile să le realizeze.

Limbajul artificial este un sistem de semne nonlingvistice, clare, care pot fi înţelese pe baza unor convenţii, la fel ca sistemul de semne dintr-o limbă naturală. Acesta este motivul pentru care a fost „creat în mod

[1] Sistemul este definit ca fiind compus dintr-o sintaxă şi o semantică, sintaxa derivând formele de suprafaţă produse de regulile de bună formare, iar semantica - o formă logică dedusă prin intermediul regulilor de compunere. Ansamblul constituit dintr-o formă de suprafaţă şi o formă logică constituie *semnificaţia frazei* (prin opoziţie cu *sensul enunţului).* Cf. Jacques Moeschler, Arme Reboul, 1999, *Dicţionar Enciclopedic de Pragmatică*. Coordonarea traducerii și prefața: Carmen Vlad, Liana Pop, Cluj-Napoca: Editura Echinox, p. 21.

[2] Interogarea este operația prin care se obțin datele dorite dintr-o baza de date, selectate conform unui anumit criteriu. Întrucât operația de interogare este cea mai importantă operație de manevrare a datelor, de multe ori limbajele de manevrare a datelor sunt denumite 'limbaje de interogare'. Deci, limbajul de interogare este un limbaj de programare specific pentru manipularea datelor în cadrul bazelor de date relaționale.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

intenționat, de unul sau mai mulţi indivizi pentru a servi la realizarea comunicării într-unul sau altul din domeniile de activitate umană”[3], după anumite reguli, în vederea fixării, prelucrării şi transmiterii de informaţii. După cum se vede, această definiţie a limbajului este valabilă atât pentru *limbajele naturale*, adică limbile formate în procesul comunicării sociale, cât şi pentru *limbajele artificiale*, construite de om în procesul cunoaşterii ştiinţifice.

Sistemul de semne este guvernat de trei tipuri de reguli:

a) *Reguli sintactice*, care vizează legăturile dintre semne.

Exemplu: *regulile legăturilor dintre diversele părţi de propoziţie.*

b) *Reguli semantice*, care vizează legătura dintre semne şi semnificaţiile lor.

Exemplu: *regulile de traducere.*

c) *Reguli pragmatice*, care stabilesc normele de utilizare a semnelor de către oameni.

Limbajul formal este „limbajul artificial cu o descriere riguroasă, matematică, bazat pe un sistem formal de tip gramatică sau automat, ce pot fi folosite ca modele ale limbajelor de programare.”[4]

Limbajul maşină este „limbajul programelor în format direct executabil al unui calculator, constituind cel mai de jos nivel pe care poate fi programat un calculator. Un program în limbajul maşină este o *secvenţă de instrucţiuni* şi *zone de date*, care pot fi imediat executate de unitatea centrală de prelucrare. Pentru reprezentările externe ale calculatorului, un program în limbajul maşină poate fi dat ca şir de cifre binare, organizate pe locaţii ale memoriei, folosind denumiri simbolice ale instrucţiunilor.”[5]

Limbile artificiale sunt limbi a căror vocabular şi gramatică au fost inventate de către un om sau un grup, spre deosebire de limbile naturale, apărute în decursul istoriei.

Limbile artificiale pot fi clasificate în:

Limbi auxiliare inventate pentru a fi folosite în comunicarea dintre oameni pe tot globul. Exemplu: *limba Esperanto*;

Limbi fictive plăsmuite în opere artistice, întâlnite în literatură sau film cu personaje imaginare;

Limbi experimentale create de lingviști ca metodă de cercetare;

Limbi secrete folosite pentru codificarea datelor;

Limbi inventate doar pentru amuzament. Exemplu: *limba „păsărească”.*

Noţiuni generale privind limbajele de programare

Calculatoarele electronice şi „limbajele de programare” joacă un rol foarte important în dialogul om-calculator.

---

[3] Cf. *Dicţionar de Informatică*, 1981, Bucureşti: Editura Ştiinţifică şi Enciclopedică, p. 193.
[4] *Ibidem*, p. 195.
[5] *Ibidem.*

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

Limbajele de programare fac parte din setul de limbaje artificiale create de om, care servesc la exprimarea, cu ajutorul calculatorului, a algoritmului de rezolvare a unei probleme, folosind o serie de comenzi executabile. Algoritmul indică modul de prelucrare, pas cu pas, a datelor primare până la obţinerea rezultatelor finale.

Un *program* este reprezentarea unui algoritm într-un limbaj de programare şi poate fi privit ca un *transformator de aserţiuni* care descrie proprietăţile datelor de intrare şi a rezultatelor.[6]

Un *limbaj de programare* este un set bine definit de caractere şi simboluri, de expresii şi reguli sintactico-semantice privind comenzile folosite pentru scrierea unui program pe un computer. Limbajul de programare este un instrument de dialog, interfața între om şi calculator.

Limbajele de programare servesc la transformarea modului de rezolvare a unei probleme într-un format accesibil calculatorului. Folosind un limbaj de programare, specialistul în informatică va crea un *soft,* care va descrie, în liniile de comandă, *calea de urmat* pentru rezolvarea unei probleme.

Limbajele de programare prin *biblioteca de comenzi*[7], pe care o oferă, dau posibilitate programatorului să specifice în programul sursă, procedurile pe care trebuie să le execute calculatorul în derularea *soft*-ului.

Toate limbajele de programare se bazează pe un set de simboluri elementare, numit *alfabetul limbajului* cu literele mari şi mici ale alfabetului latin, cu cifrele sistemului zecimal, cu anumite caractere speciale: + - * /, %....

Aceste simboluri sunt asamblate în cuvinte-cheie sau expresii care formează *vocabularul limbajului* (variabile, constante, instrucţiuni, funcţii, comenzi). Ansamblul regulilor prin care se construiesc instrucţiunile constituie *gramatica limbajului.*

Exprimarea regulilor gramaticale din limbajele de programare se realizează cu ajutorul unui *metalimbaj***.** Elementele de metalimbaj care apar în manualul de utilizare al produsului-program sunt:

- Cuvintele rezervate,[8] scrise cu majuscule şi folosite sub numele de *comenzi, clauze, funcţii.*

Exemplu: *IF(...), FOR, COPY, PRINT.*

- Cuvintele-utilizator, scrise cu minuscule şi folosite de programatori în declaraţiile de variabile şi în stabilirea câmpurilor din structura unică a unui fişier.

Exemplu: *numepren (nume prenume), cnp (codul numeric personal), tel (telefon).*

- Parantezele drepte „[ ]” încadrează o declaraţie opţională.

---

[6] *Ibidem,* p. 266.

[7] Reamintim că în metalimbajul programatic, termenul de „bibliotecă“ nu coincide cu cel din lexicul uzual, însemnând „corpus”, „inventar” de termeni, de texte etc.

[8] Prin „cuvânt rezervat” se înțelege un şir de caractere alfanumerice, care constituie un simbol cu semnificaţie particulară într-un limbaj de programare şi exprimă o comandă, o clauză etc., fără a putea fi folosit ca identificator al datelor. Cf. DI, 1981, p. 109.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

Exemplu:

Comanda SORT TO permite sortarea, ordonarea datelor într-un tabel.

SORT TO <fişier> ON <câmp1> [/A /D] [/C]
[, <câmp2>[/A /D] [/C] ...]

În sintaxa comenzii de sortare, datele din tabelul numit <fişier> vor fi aranjate după primul câmp <câmp1>, apoi după cel de-al doilea câmp <câmp2>, din structura de fişier, în ordine ascendentă/descendentă [/A|/D], sau în ordine alfabetică [/C], după literele mari sau mici ale alfabetului.

- Acoladele „{ }" sau linia verticală „|", în liniile de comandă din cadrul fişierelor program sunt alte elemente ale limbajului. Pentru o cât mai mare claritate, vom folosi un singur element: sau „{ }", sau "|".

Exemplu: *Ordonare ascendentă sau descendentă a datelor:*
[ASCENDING / DESCENDING]

Programul [9] *pentru calculator* reprezintă un ansamblu de instrucţiuni scrise cu ajutorul unui limbaj de programare. Succesiunea de instrucţiuni de pe liniile de comandă, dintr-un fişier de tip program, descriu paşii pe care trebuie să-i execute calculatorul în scopul rezolvării unei probleme.

Instrucţiunea[10] reprezintă descrierea unei acţiuni codificate, care se transmite calculatorului cu ajutorul unui limbaj de programare. Execuţia instrucţiunii, într-un interval de timp finit şi cu un efect bine definit, este realizată de către un procesor. Fiecare limbaj de programare conţine anumite tipuri de instrucţiuni: de memorare a valorilor, de atribuire, de ciclare, de control, de salt, de transfer de date de intrare/ieşire etc.

În programare, instrucţiunea de transfer prezintă forma de transfer condiţionat şi de transfer necondiţionat, folosind cuvintele-cheie IF şi THEN.

Exemplu: „*IF condiţie, THEN secvenţă, ELSE secvenţă*".
„*IF condiţie, THEN secvenţă*".

Ansamblul activităţilor de documentare, proiectare, dezvoltare, implementare şi întreţinere a programelor în elaborarea unui produs-program poartă numele de *programare*. Programul scris de un analist-programator se numeşte program-sursă. Fişierul program-sursă (notat *nume-fişier.prg*) trebuie *compilat*, moment în care calculatorul citeşte liniile de comandă şi *rulat*, transformat în format executabil (notat *nume-fişier.exe*). Formatul de tip executabil rezultă prin traducerea sursei, cu ajutorul programelor speciale numite *interpretoare* sau *compilatoare*.

---

[9] Cf. *Dicţionar de matematică şi cibernetică în economie*, Bucureşti: Editura Ştiinţifică şi Enciclopedică, 1979, s.v. *Program - Plan al procesului de prelucrare automată a informaţiei la calculatorul electronic.*

[10] Comandă dată unui calculator pentru îndeplinirea unor operaţii. Cf. Mircea Regneală, 2001, *Dicţionar explicativ de biblioteconomie şi ştiinţa informării*, vol. I. Bucureşti: FABR, p. 317.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

În cazul problemelor complexe, activitatea de creare a programelor presupune mai mult timp şi implicarea unui număr mai mare de specialişti, iar rezultatul activititǎţii este produsul-program.

Un produs-program-procedural cuprinde ansamblul de programe-procedurale cu fişierele sursă, fişierele executabile, fişierele bază de date şi *Manualul de utilizare* cu documentaţia pentru implementarea şi folosirea programului.

Manualul de utilizare accesat cu tasta F1-Help, cuprinde fişierele document cu descrierea algoritmului de rezolvare, modul de implementare şi de exploatare al *soft*-ului.

În programul-sursă printre *liniile de comandă* sunt inserate, cu ajutorul semnelor „#", * (sau alte simboluri), *liniile comentariu,* care nu influenţează descrierea structurii datelor de intrare/ieşire, a funcţiilor sau execuţia programului.

La primele limbaje de programare, trecerea de la programele sursă la programele executabile se realiza prin comenzi distincte, în care specificam ordinea de efectuare a operaţiilor. În momentul actual, programatorii se orientează către *medii de programare*.

Mediile de programare reprezintă *pachete de programe,* care asigură integrarea următoarelor funcţii: introducerea şi editarea programului sursă, interpretarea sau compilarea, editarea de legături, încărcarea şi lansarea în execuţie, plus depanarea programului. Astăzi, majoritatea limbajelor de programare sunt integrate în medii de programare având implicit instalate un editor de texte, un interpretor, un depanator de programe, care oferă fişierelor sursă o gestionare completă şi o informare rapidă prin sistemul HELP.

Alfabetul şi vocabularul unui limbaj de programare

Orice limbaj de programare are la baza două elemente:

a) Un alfabet, alcătuit din literele alfabetului englezesc, de la A la Z, majuscule şi minuscule, cu un total de 52 caractere.

b) Un set de caractere, format din cifrele arabe (de la 0 la 9, în total 10 caractere) şi caracterele speciale *(. , ; = < > # $ % + - * / " ' ( )).*

Ordonarea simbolurilor alfabetului se face pe baza codurilor numerice corespunzătoare caracterelor respective. Caracterele din alfabetul limbajului permit compunerea cuvintelor care formează vocabularul limbajului.

În limbajele de programare există următoarele categorii de cuvinte:

a) *Cuvintele cheie,* care au un înţeles clar într-un context precizat şi desemnează *nume de variabile, instrucţiuni, funcţii-condiţie etc.;*

b) *Cuvintele rezervate* ale limbajului sunt folosite doar în scopul pentru care au fost definite. Avantajele utilizării acestei categorii de cuvinte sunt următoarele:

- programul devine mai facil;
- măresc viteza de compilare a programului (citirea liniilor de program);

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

- erorile sunt depistate mai repede.

c) *Cuvintele definite de utilizator* sunt cuvinte folosite pentru a desemna diverse elemente din program: numele de câmpuri din structura de fişier, numele de funcţii, numele de proceduri, variabile, numele de fişiere.

Sintaxa şi semantica unui limbaj de programare

Dispozitivele periferice de intrare/ieșire ale unui sistem de calcul permit preluarea, prelucrarea, stocarea, transmiterea, afișarea și tipărirea datelor unui utilizator sau unui grup de utilizatori. Echipamentele periferice folosite pentru introducerea datelor în sistem sunt clasica tastatură, scanner-ul, creionul electronic etc. Cele mai folosite metode pentru transmiterea datelor sunt: poşta electronică (*e-mail*), *software* de tip *messenger* sau *chat*, *forum*-urile *electronice*, *skype* pentru video-conferinţe, reţelele de socializare, programele audio pentru Internet.

Dacă dorim să scriem o aplicaţie pentru un sistem electronic de calcul trebuie să stabilim limbajul de programare pe care îl vom folosi şi, implicit, toate comenzilor din biblioteca de comenzi a limbajului, setul de caractere, modul în care vom realiza analiza sintactică şi semantică a textului în fişierul program.

Un program este format din atomi lexicali (*tokens*) şi separatori. Un atom lexical este cea mai mică unitate sintactică cu înţeles de sine stătător într-un context precizat.

În programul sursă, analiza lexicală este acea fază a procesului de analiză care are drept scop identificarea atomilor lexicali din care e compus programul. Categoriile de *tokeni* întrebuinţaţi sunt: simbolurile speciale, identificatori, etichete şi literali.

Sintaxa unui limbaj de programare

Un limbaj de programare este un set de reguli, simboluri şi cuvinte speciale folosite pentru a scrie un program. Regulile sunt valabile atât pentru gramatică (sintaxă), cât şi pentru semantică (semnificaţie).

Cu ajutorul unui limbaj de programare şi urmând anumite reguli, vom scrie în fişierul program-sursă liniile de comandă formate din comenzi. O comandă conţine numele comenzii cu sintaxa comenzii alcătuită din cuvinte combinate, simboluri şi parametri.

Sintaxa unui limbaj de programare reprezintă un set de reguli care guvernează alcătuirea propoziţiilor dintr-un limbaj [11] . Prin sintaxă determinăm dacă o anumită instrucţiune este scrisă corect sau nu, în privinţa asocierii enunţurilor sau părţilor de enunţ.

Sintaxa limbajelor de programare este foarte *rigidă,* nu acceptă ambiguităţi şi este alcătuită dintr-un set de reguli care arată exact ce combinaţii de litere, numere şi simboluri pot fi folosite în program. Remarcăm aici o pregnantă diferenţa faţă de sintaxa limbilor naturale, unde

[11] *Cf.* DI, 1981, p. 312.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

părţile de propoziţie sau propoziţiile subordonate se pot suprima, permuta etc., desigur, între anumite limite și după anumite reguli, specifice fiecărei limbi.

Exemplu:

Eroarea de sintaxă *(syntax errors)* este generată la orice scriere incorectă şi incompletă a unei comenzi, a unui cuvânt, a unei virgule, a unui parametru de comandă.

Sintaxa unui limbaj de programare poate fi descrisă în diverse moduri, unul dintre acestea fiind notaţia BNF (Backus-Naur Form).

Notaţia BNF a fost utilizată prima dată la descrierea sintaxei limbajului ALGOL (în cadrul raportului ALGOL60, apărut în 1963). În cadrul BNF, sunt folosite metasimboluri, simboluri terminale şi simboluri neterminale.

Metasimbolurile sunt simbolurile care fac parte din mecanismul de descriere a limbajului.

Exemple de metasimboluri: *„<”, „ >”, „ ½”, „::=“*

Simbolul „½” semnifică o alternativă.

Simbolul „::=” îl vom citi „se defineşte astfel".

Simbolurile terminale sunt cuvintele care desemnează comenzile-condiţie în proceduri sau în buclele de program.

Exemplu: *FOR, WHILE, DO*.

Simbolurile neterminale sunt cuvintele încadrate între semnul mai mic „<” şi semnul mai mare „>” fiind definite în program.

Exemplu*: <variabila>, <identificator>, <instrucţiune>*.

Sintaxa unui identificator în BNF se prezintă în trei moduri alternative, folosind semnele mai mic şi mai mare.

- Un identificator este fie *o <litera>,* fie *un <identificator>* urmat de *o <cifra>,* fie *un <identificator>* urmat de o *<litera>*.

Exemplu:

<identificator>::=<litera>½<identificator><cifra>½<identificator><litera>

Unde: *<litera>::=a½b½...½z½A½B½...½Z*

<cifra>::=0½1½2½...½9

- Rezultă că un identificator poate să se identifice cu: o singură literă „*a*”*;* o literă urmată de o cifră „*b2*”*;* multe litere şi cifre „*alo2a3*”.
- Descrierea sintaxei instrucţiunii condiţionale *IF- THEN* în notaţia BNF.

Exemplu: *<instrucţiune IF>::= IF <condiţie > THEN.*

Semantica unui limbaj de programare

Semantica s-a dezvoltat ca ramură de sine stătătoare a lingvisticii abia în secolul al XX-lea. Termenul de „semantică”, introdus în literatura de

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

specialitate de către Michel Bréal[12], în 1883, a fost folosit în înţelesul actual începând cu lucrările lui Charles William Morris[13]. În cadrul semanticii, el a cercetat în special legăturile dintre semnele verbale şi ceea ce este înţeles prin ele, deosebind-o de sintaxă, care studiază relaţiile semnelor între ele, şi deosebind-o, de asemenea, de pragmatică, disciplină care se ocupă cu studiul relaţiilor dintre semne şi utilizatorul lor. Lingvistica structuralistă a preluat această diferenţiere şi a dezvoltat mai departe studiul semnificaţiei simbolurilor verbale. În cadrul semanticii, studiul semnificaţiei cuvintelor se numeşte *semasiologie,* iar studiul denumirilor, *onomasiologie*, termeni care se regăsesc numai parţial în limbajele de programare.

- *Sensul* din aceste limbaje artificiale se referă la legăturile dintre cuvintele, semnele şi parametri unei comenzi.
- *Semnificaţia* cuvintelor se referă la relaţia dintre semne cu întreg programul.

O preocupare importantă a semanticii, tradiţională, de altfel, o constituie transformarea în formule logice a exprimărilor naturale într-o anumită limbă, după o metodă care a fost dezvoltată de Richard Montague. Spre deosebire de fonetică, morfologie şi sintaxă, obiectul de studiu al semanticii *„... est l'étude du sens des mots, des phrases et des énonces*"[14].

Domeniile de cercetare ale semanticii din limbajele de programare coincid, în bună parte, cu cele din lingvistica generală aplicabilă limbilor naturale.

Domeniile de cercetare ale semanticii sunt[15]:

- *Semantica lexicală* are ca obiect studiul semnificaţiei cuvintelor şi al structurii interne a vocabularului în întregime.
- *Semantica propoziţională* cercetează felul în care, din sensul fiecărui cuvânt în parte rezultă unităţi sintactice mari (fraze) cu semnificaţie proprie.
- *Semantica textelor* analizează combinaţia propoziţiilor reale sau ipotetice, din care rezultă o descriere, o naraţiune sau o argumentaţie coerentă.
- *Semantica discursivă* are ca obiect studiul exprimărilor diverselor persoane care se găsesc angajate într-o discuţie, convorbire banală sau într-o dispută ştiinţifică.

Relaţiile între limbă şi gândire, respectiv între limbă şi lumea existentă ne permit să amintim şi de alte domenii ale semanticii:

- *Semnificaţia cognitivă* se referă la relaţia dintre limbă şi gândire. O exprimare verbală poate fi înţeleasă doar în măsura în care ea reconstruieşte structura gândirii celui ce o exprimă.

---

[12] Michel Bréal, 1883/2012, *Essai de Sémantique*, Paris: Hachette.
[13] Charles William Morris, 1949, *Foundations of the Theory of Signs*, Chicago: University of Chicago Press.
[14] Irène Tamba-Mecz, 1988, *La sémantique,* Paris, PUF, p. 7.
[15] http://ro.wikipedia.org/wiki/Semantic%C4%83.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

- *Semnificaţia informaţională*, denumită şi *teorie referenţială*, este disciplina conform căreia rolul principal îl joacă relaţia dintre limbajul descompus în unităţi informaţionale şi persoana referentă.
- *Semnificaţia pragmatică* studiază relaţiile între sensul lingvistic al unei exprimări şi un anumit context.

În cazul limbajelor de programare, semantica reprezintă un set de reguli care determină semnificaţia instrucţiunilor scrise într-un fişier program.

Semantica comenzii, de pildă, arată sensul cuvintelor dintr-o comandă, având ca scop clarificarea sensului şi a semnificaţiei noţiunilor complexe, derivate din simbolurile cele mai simple ale limbajului, sprijinindu-se pe regulile sintaxei, fără a se identifica însă cu aceasta.

În logică şi informatică se folosesc termenii de „*Semantică formală*" sau „*Semantica limbajului de programare*". Între semantică şi sintaxă există acelaşi raport ca între fond şi formă.

Analiza textului într-un program sursă

Analiza textului-sursă, adică a liniilor de comandă din programul-sursă, constă în: analiza lexicală, analiza sintactică şi analiza semantică a acestuia.

Analiza lexicală este prima *operaţie* pe care o facem asupra *fişierului program-sursă*, considerat un şir de caractere care conţine subşiruri de caractere, numite „atomi lexicali", cu operatori (+,-,<,>), cuvinte-cheie sau cuvinte rezervate, constante (10,1,2,3, 23, A, ANA), identificatori şi separatori.

Analizorul lexical execută următoarele operaţii:

- detectează în programul-sursă subşirurile care respectă regulile de formare a atomilor lexicali;
- clasifică subşirurile, adică identifică clasa de care aparţin aceste subşiruri;
- traduce subşirurile în atomi lexicali;
- memorează atomii în tabela de simboluri.

Analiza sintactică a şirului atomilor lexicali identifică structurile sintactice - expresii, liste, instrucţiuni, proceduri - şi generează o descriere structurală a acestuia. În cazul în care şirul de intrare este corect sintactic, apare *arborele sintactic* (de derivare) sau, în caz contrar, un *mesaj de eroare*. Arborele sintactic descrie relaţiile de separare ori de incluziune dintre structuri.

Analiza semantică foloseşte arborele sintactic, creat în faza de analiză sintactică, pentru a extrage informaţii privind apariţiile în programul-sursă a obiectelor purtătoare de date (tipuri de date, variabile, proceduri, funcţii) şi pentru a atăta necesitatea utilizării lor. Odată cu parcurgerea arborelui sintactic are loc şi generarea codului intermediar. Acesta reprezintă un şir de instrucţiuni simple, cu format fix, în care codurile operaţiilor sunt asemănătoare cu codurile maşină corespunzătoare, ordinea operaţiilor respectă ordinea execuţiei (conform apariţiei lor în

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

programul sursă), iar operanzii sunt prezentaţi sub forma variabilelor din programul-sursă (nu sub forma de adrese de memorie).

În practică, analiza semantică are loc în paralel cu cea sintactică, prin asocierea acţiunilor analizorului sintactic cu anumite structuri de date ce reprezintă atribute ale componentelor sintactice.

Generarea codului obiect presupune alocarea locaţiilor de memorie şi a *registrelor* [16] unităţii centrale pentru variabilele programului şi înlocuirea codurilor de operaţii din codul intermediar cu codul maşină.

Componentele de bază (lexicală, sintactică, semantică) sunt asistate pe tot parcursul compilării de următoarele două module:

a) *Modulul de tratare a erorilor* prezintă o colecţie de proceduri care sunt activate ori de câte ori este detectată o eroare în timpul operaţiilor de analiză.

În funcţie de *faza de analiză* în care se află programul, erorile pot fi *lexicale, sintactice* sau *de semantică*. Modul de tratare a erorilor afişează mesajele de diagnostic relativ la eroarea depistată şi iau decizii privind metoda de continuare a traducerii: fie se continuă compilarea, ignorând elementul ce conţine eroarea, fie se încearcă corectarea erorii sau se întrerupe traducerea.

b) *Modulul de gestiune a tabelelor* prezintă o colecţie de proceduri care creează şi actualizează baza de date a compilatorului şi conţine informaţii proprii compilatorului (generate la implementare şi constituite din mecanismele de descriere a analizei lexicale, sintactice şi semantice) şi informaţii ce aparţin programului–sursă, care se traduce (identificatorii, constantele, cuvintele-cheie), memorate în tabela de simboluri.

Gestionarea tabelei de simboluri se constituie în funcţie de: modul de reprezentare al acesteia, caracterul general al tabelei, tipul limbajului din care se traduce programul-sursă şi convenţiile alese pentru reprezentarea atributelor. De obicei, în faza de analiză lexicală, la întâlnirea unui nume nou, acesta este întrodus în tabela de simboluri, reţinându-se şi adresa intrării. Orice referire ulterioară la acest nume actualizează informaţia din tabela corespunzătoare acestui nume, verificându-se şi consistenţa utilizării acestuia (în cadrul analizei semantice). La generarea codului, atributele numelui determină lungimea zonei de memorie alocată acestuia. Atributele numelui pot servi şi în faza de tratare a erorilor.

Translatorul programelor de calculator

Translatorul este programul care traduce fişierele-sursă scrise într-un anumit limbaj de programare, într-un program echivalent dintr-un alt limbaj, acesta numindu-se „program destinaţie”.

Translatoarele se clasifică în:

[16] Registrul este un dispozitiv destinat memorării unui vector binar. Registrul de adrese păstrează adresa curentă folosită pentru efectuarea accesului la o memorie. *Cf.* DI, 1981, p. 287.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

- *Compilatoare* - pentru acestea programul destinaţie se numeşte *program obiect* sau *cod obiect*, fiind apropiat de *codul maşină*. În cazul limbajelor de nivel înalt, translatorul poartă denumirea de *compilator sau interpretor*.
  - Compilatoarele sunt translatoare care citesc tot programul scris în limbajul cod sursă şi îl traduc în *cod maşină*.
  - Interpretoarele sunt translatoare care citesc pe rând fiecare instrucţiune din programul sursă, după care o traduc şi o execută.
- *Asambloare* – sunt compilatoarele limbajelor de asamblare folosite pentru traducerea mnemonicilor din limbajul de asamblare în limbajul maşină.

În aceste două cazuri, traducerea este urmată, de obicei, de editarea de legături, înainte de execuţia propriu-zisă a programului. În această fază, codul executabil se constituie prin legarea codului-obiect rezultat din traducere cu alte module obiect (rezultate ale unor compilări anterioare sau cele existente în bibliotecile de comenzi).

- *Interpretoarele* realizează execuţia programului sursă instrucţiune cu instrucţiune.
- *Preprocesoarele* traduc programele-sursă din limbaje de nivel înalt în programe destinaţie tot în limbaje de nivel înalt.
- *Cross-compilatoarele sau cross-asambloare* generează pe un calculator „gazdă" un cod obiect pentru un alt calculator-obiect (care are memorie mică şi nu poate implementa programul de traducere);
- *Compilatoarele incrementale* reprezintă o combinaţie între compilator şi interpretor, care foloseşte secvenţe din programul-sursă cu o anumită independenţă sintactică şi semantică pentru a le executa interpretativ.

Execuția limbajelor de programare

Pentru executarea unui program scris într-un limbaj oarecare, există, în principiu, două abordări: compilare sau interpretare. La compilare, compilatorul transformă programul-sursă, în totalitatea sa, într-un program echivalent scris în limbaj maşină, care apoi este executat.

La interpretare, interpretorul ia prima instrucţiune din programul-sursă, o transformă în limbaj maşină şi o execută; apoi trece la instrucţiunea a doua şi repetă aceleaşi acţiuni. Unele limbaje se pretează bine la compilare; de exemplu, limbajele clasice: Pascal, C, C++, altele sunt interpretate, de exemplu, SQL, PHP.

Multe limbaje moderne combină compilarea cu interpretarea: codul sursă este *compilat* într-un limbaj binar numit *bytecode*, care la rulare este *interpretat* de către o *maşină virtuală*. De remarcat este faptul că unele interpretoare de limbaje pot folosi compilatoare aşa-numite *just-in-time*, care transformă codul în limbaj maşină chiar înaintea executării.

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

Bibliografie

BÂRLEA, Petre Gheorghe, 2015, „Pentru o analiză semantică informatizată a textului poetic", în: Florina Loredana Streinu, *Poezia Anei Blandiana - o analiză semantic-textuală*, Bucureşti: Editura Muzeul Literaturii Române, pp. 5-10.

BÂRLEA, Petre Gheorghe, 2014, „Baza logică a structurilor condiționale în limbile naturale", în: Gabriela Duda (coord.), *Cultura limbii. Omagiu Doamnei Profesor Domnița Tomescu*, Ploiești: Editura Universității de Petrol și Gaze, pp. 53-64.

BRÉAL, Michel, 1883/2012, *Essai de Sémantique*, Paris: Hachette.

Dicţionar de Informatică (DI), 1981, Bucureşti: Editura Ştiinţifică şi Enciclopedică.

Dicționar de matematică și cibernetică în economie, București: Editura Științifică

și Enciclopedică, 1979.

MOESCHLER, Jacques; Reboul, Arme, 1999, *Dicţionar Enciclopedic de Pragmatică*, Cluj-Napoca: Editura Echinox.

MORRIS, Charles William, 1949, *Foundations of the Theory of Signs*, Chicago:University of Chicago Press.

REGNEALĂ, Mircea, 2001, *Dicționar explicativ de biblioteconomie și știința informării*, vol. I. București: FABR.

TAMBA-MECZ, Irène, 1988, *La sémantique,* Paris: PUF.

BDD-A24402 © 2016 Ovidius University Press

Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)

BDD-A24402 © 2016 Ovidius University Press
Provided by Diacronia.ro for IP 216.73.217.31 (2026-11-26 23:04:38 UTC)