Bogdan PRICOP
Gheorghe BĂDĂRĂU

Cap 3. Elemente de programare

- ***Algoritmi***
- ***Limbaj cod-mașină, limbaj de asamblare***
- ***Limbaje de programare. Etapele dezvoltării unui program***
- ***Limbajul Fortran 90***
- ***Convenții utilizate***

Algoritmi

Procesul elaborării de către om a programelor în scopul comunicării cu computerul pentru rezolvarea unor probleme se numeşte ***programare***. Programarea începe întotdeauna cu o *definiție* clară a problemei de rezolvat, adică un enunț cât mai precis, cu specificarea datelor de intrare, datele pe care le cunoaștem, și a datelor de ieșire, a rezultatelor pe care dorim să le aflăm. După ce problema a fost definită, pentru rezolvarea ei, alegem un algoritm.

Mai precis, un ***algoritm*** este o rețetă, formată dintr-un număr finit de pași de prelucrare, rețetă ce descrie procesul de transformare a datelor de intrare pentru a obține datele de ieșire.

Algoritmi

Orice algoritm trebuie să posede următoarele trei proprietăți :

- În primul rând, ***algoritmul trebuie să fie definit****;* fiecare pas de prelucrare este descris foarte precis și riguros, astfel încât pentru cel ce execută algoritmul să nu existe nimic de neînțeles și să nu apară nici un fel de ambiguitate în interpretare.
- În al doilea rând, ***algoritmul trebuie să fie eficace****;* în toate cazurile pentru care a fost creat, algoritmul conduce la rezultat după un număr finit de pași de prelucrare relativ simpli. Acesta proprietate se mai numește *realizabilitate potențială*.
- În al treilea rând, ***algoritmul trebuie să aibă caracter de masă****;* să permită rezolvarea oricărei probleme ce aparține unei anumite clase de probleme.

Limbaj cod-mașină, limbaj de asamblare

Limbajul algoritmic înțeles de către un computer se numește ***limbaj cod-maşină***, ***limbaj obiect*** sau ***limbaj intern***. Este un limbaj binar în care propozițiile limbajului se scriu numai cu simbolurile 0 și 1. În funcție de tipul procesorului fiecare computer, posedă un număr finit de instrucțiuni simple, *comenzi,* pe care le poate executa. Oricât de diferite ar fi comenzile de la un computer la altul, ele prezintă unele caracteristici generale. Orice comandă conține un *cod al operației* ce trebuie executată şi de obicei o *adresă*.

Codul operației poate indica: adunare, scădere, înmulțire, împărțire, comparație, citire din memorie, depunere în memorie, salt condiționat sau necondiționat, etc.

Adresa indică locul de unde se ia informația ce trebuie prelucrată sau locul unde trebuie depus rezultatul (memorie sau registre).

Limbaj cod-mașină, limbaj de asamblare

Pentru a uşura munca de scriere a unor programe s-au inventat *limbajele de asamblare*, în care codul operației se reprezintă printr-o mnemonică (*se referă la memorie; care ajută memoria*), iar adresele binare se înlocuiesc cu adrese simbolice. Programul sursă scris în limbajul de asamblare este tradus în mod automat de computer în limbajul obiect (limbajul binar) cu un program special numit *assembler;* Procesul de traducere poartă numele de ***asamblare***.

Până la sfârșitul anilor '50 programele se scriau doar în limbajele de asamblare sau în limbajele cod-maşină. Această activitate se numește de obicei *codificare,* iar persoana ce o practică se numește *codificator*.

Cu apariția unor computere mai rapide și cu memorie mai mare, dificultățile de codificare au crescut și a devenit evident că nu este rațional ca această muncă obositoare de codificare să o facă omul.

Limbaje de programare

Tendințele de a înlătura insuficiențele limbajelor de asamblare şi ale limbajelor cod-maşină au condus la crearea *limbajelor de programare,* sau cum se mai spune, a *limbajelor de programare de nivel înalt*.

Cu ajutorul acestor limbaje de programare, *procesul dezvoltării unui program* a fost redus la următorii patru pași:

- *Pasul* 1 - ***Crearea și editarea***. Programatorul lansează în execuție la computer un program numit *editor de texte* și redactează de la tastatură un *fișier sursă* ce conține textul programului scris în limbajul de programare. Crearea și editarea se pot face separat; mai întâi programatorul își concepe programul folosind hârtie și creion și apoi, cu editorul de texte, îl rescrie de la tastatură creând fișierul sursă. Sau, programatorul, pur și simplu, concepe programul și în același timp îl și editează.

Limbaje de programare

- *Pasul 2*. La computer se lansează în execuție ***compilatorul***. La intrare, compilatorul are fișierul sursă și la ieșire se obține fișierul *obiect* cu rezultatul traducerii, adică programul în limbaj binar. Fișierul obiect se obține doar în cazul în care toate instrucțiunile fișierului sursă au respectat întocmai regulile sintactice ale limbajului de programare. Dacă în fișierul sursă există greșeli de sintaxă, compilatorul generează doar un fișier cu o listă de erori de compilare.
- *Pasul* 3. La computer se lansează în execuție ***editorul de legături*** (linker). Linker-ul combină unul sau mai multe fișiere obiect cu fișiere obiect din bibliotecile computerului și generează *programul executabil*.
- *Pasul* 4. La computer se lansează în execuție programul executabil. Dacă nu există erori programul generează fișierul (fișierele) *cu* ***rezultate*** .

Limbajul Fortran 90

Primul limbaj de programare care a fost standardizat este limbajul Fortran. Cuvântul *FORTRAN* este un acronim de la cuvintele englezești "**FOR**mula **TRAN**slation" ceea ce înseamnă "traducător de formule". Limbajul Fortran a fost conceput astfel încât să fie convenabil rezolvării problemelor științifice inginerești, probleme în care apar *formule matematice*. **Fortran** este un limbaj de programare născut în anul 1950 și care este încă folosit după mai bine de jumătate de secol de existență.

De-a lungul timpului, acest limbaj de programare a fost în continuu îmbunătățit ajungându-se la varianta **Fortran 2015** ce va fi lansat la jumătatea anului 2018.

Convenţii utilizate

Pentru a descrie în mod economic şi precis sintaxa limbajelor de programare se foloseşte de obicei un ***metalimbaj***, un limbaj de descriere a limbajelor. În metalimbaj propoziţii ale limbajelor naturale se înlocuiesc cu nişte notaţii, numite *metasimboluri*. Cele mai des folosite sunt metasimbolurile introduse de Backus şi Naur. Pentru prezentarea limbajului Fortran 90 vom folosi notaţii Backus-Naur modificate.

Facem următoarele convenţii:

1. Cuvintele scrise cu caractere mari cu fontul ARIAL desemnează cuvinte cheie cu o semnificaţie specială pentru limbajul Fortran. Ele sunt o parte componentă a instrucţiunilor în afară de cazul în care aceste cuvinte sunt închise între paranteze pătrate [ ].

Convenții utilizate

2. Literele mici şi cuvintele scrise cu caractere mici *italice* (adesea prescurtat) reprezintă clase sintactice pentru care entitățile sintactice specifice trebuie substituite cu informația dată de utilizator.

3. Metasimbolurile folosite sunt următoarele:

[ ]	- un element opțional.
[ ]...	- include un element opțional repetat; poate să apară de zero sau mai multe ori.
este	- introduce o definiție a unei clase sintactice.
{a1|a2}	- indică o alegere între elementele a1 și a2
▪	- continuă o regulă sintactică

4. Caracterul blanc (spațiu liber) se reprezintă cu caracterul *b* (italic).

Convenții utilizate

Exemplu

constantă_întreagă este *cif* [*cif*] ...

unde *cif* este o cifră. De aici rezultă că o constantă întreagă se poate scrie:

cif

cif cif

cif cif cif cif

cif cif cif cif cif

Dacă înlocuim *cif* cu constante concrete, atunci exemplu va arăta așa:

3

29

12345

Bibliografie

- *Octavian PETRUȘ, Fortran 90/95, Limbaj și Tehnici de programare, Editura Universității Tehnice "Gheorghe Asachi" din Iași, 2001*
- Romeo CHELARIU, Sisteme de operare şi limbaje de programare (Îndrumar de laborator), http://www.sim.tuiasi.ro/wp-content/uploads/Chelariu-indrumar-solp.pdf, 2004
- https://ro.wikipedia.org