Nicolae Constantinescu

Nicolae Constantinescu

Bazele Programării Procedurale

Limbajul *C*

Ediția a IV-a adăugită și revizuită

Editura UNIVERSITARIA
Craiova, 2018

Referent ştiinţific:

Prof. univ. dr. Ion Iancu

Descrierea CIP a Bibliotecii Naţionale a României
CONSTANTINESCU, NICOLAE
Bazele programării procedurale / Nicolae Constantinescu. - Ed. a 4-a, adăug. şi rev.. - Craiova : Universitaria, 2018
Conţine bibliografie
ISBN 978-606-14-1442-0

004

Dedicație

mamei mele, Victoria

Prefață

Primii pași în programare te fac să îndrăgești acest lucru sau să îți fie teamă de el. Teama vine din neînțelegere, din frica de a încerca iarăși și iarăși acele compilări care duc la nesfârșite pagini de erori, pentru cei care nu pătrund tainele programării. Când înțelegem, totul devine simplu și se așterne zâmbetul.

Dacă cititorul se va așeza și va citi cu mult (subliniez: mult) calm acest manuscris, dacă va introduce toate exemplele într-un limbaj de programare și va rula pas cu pas, la sfârșit îi pot garanta un lucru: i se va așterne zâmbetul.

Vă doresc ca din acest zâmbet să se nască o carieră în ceea ce se numește programarea sistemelor de calcul, prin care tot mai mult din ceea ce ne înconjoară devine un șir de date stocate, prelucrate, transmise.

Când spun programare nu înseamnă că vorbesc numai de clasicul computer, mă refer la tot ce conține un chip și un program în el.

Vreau să adaug un lucru: doi dintre colegii voștrii m-au ajutat cu adevărat corectând unele erori sau oferind alte soluții, mai apropiate de înțelegerea voastră, la unele programe. Numele lor este Alin Golumbeanu și Tiberiu Alboiu. Le mulțumesc și le doresc o carieră în domeniu care să le aducă acel zâmbet al realizarii unui lucru serios și dincolo de demagogia existentă în acest mediu.

Succes!

Capitol 1

Noțiuni introductive

1.1 Vedere de ansamblu - limbajul *C*

Limbajul *C* a fost inventat și implementat de Denis Ritchie pentru sistemul de operare *UNIX*. Foarte multe idei din *C* sunt preluate din limbajul *BCPL*, care a stat la baza programului numit *B*. Odată cu dezvoltarea calculatoarelor personale a crescut răspândirea și dezvoltarea limbajului *C*.

C este încadrat în categoria limbajelor de programare de nivel mediu. El combină cele mai reușite elemente ale limbajelor de nivel înalt (*FORTRAN*, *PASCAL*) cu flexibilitatea și gradul de control oferite de limbajul de asamblare. *C* permite manevrarea biților, a octeților și a adreselor de memorie. De asemenea este un limbaj portabil. Portabilitatea semnifică ușurința de a adapta programele scrise pentru un anumit sistem de operare sau un anumit tip de calculator, la alte sisteme de operare, respectiv calculatoare. Datorită portabilității, *C* a devenit din ce în ce mai popular.

Limbajul *C* include 5 tipuri de date, cele mai cunoscute fiind tipul caracter, tipul întreg și real. Un tip de date se definește ca fiind un set de valori stocate într-o variabilă sau constantă, putând să fie efectuate un set de operații asupra lor. În comparație cu limbajul *PASCAL*, *C* nu este un limbaj puternic tipizat, el permițând

aproape toate tipurile de conversii (vezi 2.6).

Un avantaj al limbajului *C* este numărul mic de cuvinte cheie, 32. Un limbaj de nivel înalt are de câteva ori mai multe cuvinte cheie (de exemplu *BASIC* are peste 100 de cuvinte cheie).

O altă deosebire față de limbajele de nivel înalt o reprezintă depistarea erorilor. *C* nu execută depistarea anumitor tipuri de erori în timpul rulării. În grija programatorului revine și verificarea limitelor unui tablou (vector, matrice). *C* permite ca tipul unui argument să fie diferit de tipul parametrului care va primi argumentul (se realizează conversii). În limbajele de nivel înalt acest lucru nu este posibil, fiind necesară o compatibilitate strictă între cele 2 tipuri.

Un limbaj structurat este un limbaj ce permite compartimentarea codului și a datelor. *C* este un limbaj structurat: el are proprietatea de a separa o anumită secvență de cod și de a o ascunde față de restul programului. Principala componentă de structură a unui program *C* o reprezintă funcția. Ea permite executarea unor secvențe de instrucțiuni fără a crea efecte secundare în altă parte a programului. Funcția reprezintă o caracteristică importantă a limbajului *C*, deoarece în cadrul proiectelor de anvergură mare codul scris de un programator nu trebuie să afecteze, nedorit, codul scris de un alt programator. O altă metodă de structurare a codului *C* o reprezintă blocurile de cod. Un bloc de cod este un grup de declarații/instrucțiuni pe care programul le vede ca pe un întreg. Un bloc de cod se crează prin inserarea între acolade, una deschisă, alta închisă, a unui set de declarații/instrucțiuni.

Exemplu:

```
1 if(i>0){
2     printf("Numarul␣%d␣este␣nenul", i);
3     --i;//decrementare i; analog i=i-1
4 }
```

Alte proprietăți ale limbajului ce au sporit popularitatea acestuia printre pro-

gramatori sunt numărul mic de restricții, mesaje de eroare limitate, un număr redus de cuvinte cheie și posibilitatea de a crea structuri de tip bloc, funcții de sine stătătoare.

Compilatoarele de *C* tind să producă la ieșire coduri obiect foarte compacte și foarte rapide. *C* permite crearea și întreținerea programelor foarte mari cu un efort redus, deoarece permite compilarea separată și crearea și întreținerea bibliotecilor de funcții specifice fiecărui programator.

Bjarne Stroustrup a creat limbajul *C++*, adăugând un nivel de abstractizare peste limbajul *C*. Limbajul *C* are mari îmbunătățiri în comparație cu limbajul de asamblare, dar el cere totuși ca problema să fie gândită în termenii de structură ai calculatorului mai degrabă decât din punct de vedere al structurii problemei ce necesită rezolvarea. Limbajul *C++* rezolvă această problemă, abordând programarea orientată pe obiecte într-o manieră îmbunătățită față de limbajele *Simulla* și *Smalltalk*. *OOP* iți permite să descrii problema în termenii ei, mai degrabă decât în termenii calculatorului unde va rula soluția. Limbajul *C++* conține și acceptă limbajul *C* în totalitate alături de tehnica programării orientate pe obiect. Limbajul *C++* a fost conceput pentru tehnica programării orientate pe obiect dar datorită faptului că este dezvoltat din *C*, încorporează și principiile programării structurate.

Alte limbaje de programare dezvoltate pornind de la limbajul *C* sunt: *Perl*, *Java*, *Javascript* și *C#*.

1.2 Etapele realizării unui program

Realizarea unui program pentru o anumită problemă presupune parcurgerea mai multor etape. Aceste etape sunt independente de limbajul de programare utilizat și implică existența câtorva restricții cu privire la computerul utilizat.

Etapele realizării unui program sunt următoarele:

1. Studierea detaliată a cerințelor aplicației. Este foarte important ca cerințele impuse de aplicație să fie foarte bine explicate. Adică înainte de a trece la realizarea unui program pentru o anumită aplicație trebuie ca acea aplicație să

fie foarte bine analizată și cerințele impuse de aceasta trebuie să fie complete și consistente. De exemplu o cerință de genul "scrie un program care să rezolve ecuațiile" este incompletă. Se impun întrebări de genul "ce tip de ecuații", "câte ecuații", "care este precizia", etc.

2. Analiza problemei și determinarea rezolvării acesteia. Adică în această etapă se decide asupra unei metode care poate să rezolve problema. O astfel de metodă este des denumită *ALGORITM*.

3. Traducerea codului sursă al unui program în cod mașină, pe care calculatorul îl poate citi și executa nemijlocit. Textul unui program ce poate fi citit de către utilizator reprezintă codul sursă (*source program* sau *source code*). În această etapă programul trebuie citit și verificat pentru a i se stabili corectitudinea. Aceasta se face prin introducerea unui set de valori și testarea dacă programul furnizează valorile corespunzătoare corecte. Odată verificat programul este introdus în computer prin intermediul unui Editor.

4. Compilarea programului în limbajul mașină. Astfel programul obținut în limbaj mașină se numește cod obiect. În această etapă compilatorul poate determina erori de sintaxă ale programului. O eroare de sintaxă este o greșeală în gramatica limbajului. De exemplu *C* necesită ca fiecare linie să se termine cu un caracter **;**. Dacă se uită plasarea **;** atunci compilatorul va semnala eroare de sintaxă. Astfel se repetă compilarea până la eliminarea tuturor erorilor de sintaxă.

5. Programul obținut în urma compilării, codul obiect, este apoi corelat (*linked*) cu o serie de biblioteci de funcții (*function libraries*) furnizate de sistem. Toate acestea se petrec cu ajutorul unui program numit *linker* (un program ce leagă funții compilate separat într-un program unic). Programul combină funcțiile aflate în biblioteca standard *C* cu unele scrise de programator, iar apoi programul *linked object code* este încărcat în memoria computerului de către un program numit *loader*. Datele de ieșire ale unui *linker* alcătuiesc un program executabil.