Ministerul Educaţiei şi Cercetării

Informatică

Profil real

Specializările:
matematică-informatică
ştiinţe ale naturii

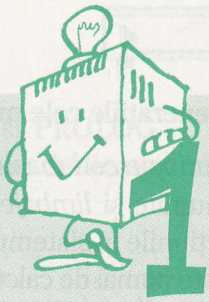
Mioara Gheorghe
(coordonator)
Constanţa Năstase
Monica Tătărâm

Manual pentru clasa a X-a

CORINT

CUPRINS

LIMBAJE DE PROGRAMARE — ELEMENTE DE BAZĂ



1. NOȚIUNI INTRODUCTIVE

1.1. Evoluția limbajelor de programare

În clasa a IX-a, s-au construit algoritmi pentru rezolvarea unor probleme din diverse arii de activitate (matematică, fizică, chimie etc.). Algoritmii au fost reprezentați prin secvențe pas cu pas sau în pseudocod. Pentru a prelucra un algoritm prin intermediul unui sistem de calcul, algoritmul trebuie descris într-un limbaj de programare.

Rețineți!

- ***Limbajul de programare*** reprezintă un mijloc de comunicare între utilizatorul uman, care este programatorul, și sistemul de calcul.
- Descrierea algoritmului în limbaj de programare se face cu ajutorul unui ***program***.
- Un ***program*** este o succesiune de comenzi — ***instrucțiuni*** ce vor fi executate de sistemul de calcul.
- Un calculator poate să „înțeleagă" mai multe limbaje de programare întrucât fiecare limbaj are un „traducător" — ***compilator*** propriu.

Evoluția limbajelor de programare a avut loc în paralel cu evoluția sistemelor de calcul (figura 1).

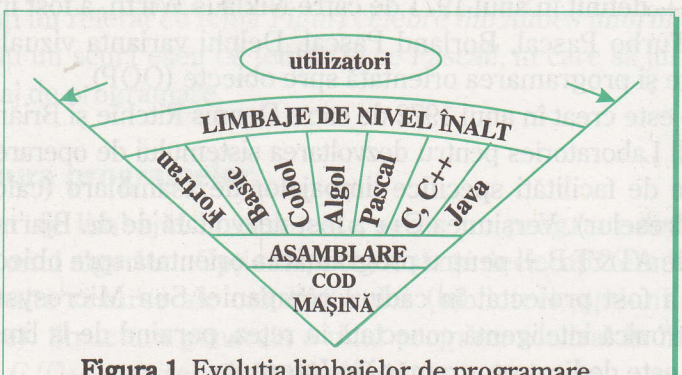


Figura 1. Evoluția limbajelor de programare

1

♦ Generațiile cele mai importante ale limbajelor de programare

1. *Limbaje cod-mașină*

Denumite și *limbaje de bază* sau *de nivel zero*, ***limbajele cod-mașină*** descriu instrucțiunile în sistemul de numerație binar (secvențe de 1 și 0). Programele sunt executate numai de calculatorul pentru care au fost scrise.

Primul program a fost realizat pentru mașina mecanică a lui Charles Babbage (1834) de către contesa Ada Lovelace, fiica poetului Lord Byron.

2. *Limbaje de asamblare*

Limbajele de asamblare au la bază un set de coduri (*mnemonice*) care sunt reprezentări simbolice ale instrucțiunilor mașină. Un program specializat, **asamblorul**, translatează aceste coduri în sistemul binar, astfel încât să poată fi decodificate și prelucrate de procesorul calculatorului. Fiecare tip de procesor are un limbaj de asamblare propriu.

3. *Limbaje de nivel înalt*

Limbajele de programare de nivel înalt sunt mai apropiate de limbajul natural în care gândim și comunicăm noi. Aceste limbaje folosesc cuvinte din vocabularul limbii engleze, sunt accesibile și au o arie largă de aplicație: calcule științifice sau economice, reprezentări grafice, probleme de optimizare, jocuri.

Cele mai reprezentative limbaje de nivel înalt sunt:

- **FORTRAN** (**FOR**mula **TRAN**slation) — a apărut în anul 1955, fiind destinat calculelor tehnico-științifice.
- **COBOL** (**CO**mmon **B**usinesss **O**riented **L**anguage) — a apărut în anul 1960; limbajul este orientat spre rezolvarea problemelor economice.
- **BASIC** (**B**egginer's **A**llpurpose **S**ymbolic **I**nstructions **C**ode) — limbajul a fost conceput în anul 1964, impunându-se puternic în perioada 1975-1980. Variantele realizate mai recent (Quick Basic, Visual Basic) sunt utilizate cu succes pentru dezvoltarea unor aplicații complexe.
- **PASCAL** — definit în anul 1971 de către Niklaus Wirth, a fost îmbunătățit în noi variante: Turbo Pascal, Borland Pascal, Delphi varianta vizuală. Versiunea actuală permite și programarea orientată spre obiecte (OOP).
- **C/C++** — este creat în anul 1972 de către Dennis Ritchie și Brian Kernigham de la firma Bell Laboratories pentru dezvoltarea sistemului de operare Unix. Acest limbaj dispune de facilități specifice limbajelor de asamblare (calculul pe biți, prelucrarea adreselor). Versiunea C++ a fost dezvoltată de dr. Bjarne Stroustrup în laboratoarele AT&T Bell pentru programarea orientată spre obiecte.
- **JAVA** — a fost proiectat în cadrul companiei Sun Microsystems pentru aparatură electronică inteligentă conectată în rețea, pornind de la limbajul C/C++; limbajul JAVA este dedicat programării în Internet.

• **LISP** (LISt Processing Language), creat în 1965, și **PROLOG** (**PRO**gramming **LOG**ic), creat în 1973 — sunt limbaje dedicate rezolvării problemelor de inteligență artificială.

♦ Stiluri de programare

Evoluția limbajelor de programare a determinat formarea mai multor stiluri de programare. **Stilul de programare** reflectă atât modul de gândire al programatorului, cât și felul în care acesta descrie algoritmul la nivel de program.

1. **Programarea nestructurată** — stil „liber" de programare, fără reguli; din acest motiv, programele nestructurate au un aspect „dezordonat", fiind mai greu de urmărit și de depanat. Acest stil de programare este specific programatorilor care folosesc limbajele de programare FORTRAN, BASIC.

2. **Programarea structurată** — stil de programare care respectă principiul: *„orice program poate fi implementat doar prin structuri de control secvențiale, alternative sau repetitive"* (teorema de structură Böhm și Jacopini). Programele structurate pot fi realizate doar în limbajele de programare care au instrucțiuni echivalente structurilor de control. Pascal și C/C++ sunt astfel de limbaje.

3. **Programarea orientată spre obiecte (OOP)** — tendință nouă de programare care îmbină programarea structurată cu tehnica descrietii datelor și a prelucrărilor prin analogie cu obiectele din lumea reală. Un obiect este descris prin caracteristici și funcții, poate proveni din alt obiect sau poate genera, prin transformare, un obiect nou. Limbajele de programare Pascal și C/C++ au și versiuni OOP.

În acest manual, sunt prezentate elemente de bază ale programării structurate utile unui programator începător, cu exemplificări în limbajele Pascal și C/C++.

TEME

1. Realizați o scurtă prezentare a limbajelor de programare urmărind evoluția în timp a acestora.
2. Realizați un referat cu tema *Figuri celebre din lumea informaticii.*
3. Realizați un scurt eseu cu tema *De ce Pascal!*, în care să justificați numele acestui limbaj de programare.

1.2. Structura programelor

Indiferent de limbajul în care este scris, un program descrie datele și prelucrările unui algoritm. Opțional, pot exista și declarații tehnice prin care se solicită anumite resurse ale calculatorului (biblioteci, opțiuni de compilare, preprocesare). Structura generală a unui program realizat în limbajul Pascal, respectiv, în C/C++ este redată în tabelul 1.

Tabelul 1

Structura programelor

LIMBAJUL PASCAL	LIMBAJUL C/C++
program identificator_program; *declarații opțiuni de compilare;* {$} *declarații de* **UNIT**-*uri;* (fișiere bibliotecă) **uses** crt, graph, dos; *definiții de constante;* **const** n=15; *definiții de tipuri de date* **type** sir=**array**[1..5]**of real**; *declarații de variabile;* **var** x, y : byte; *declarații de subprograme* (funcții și/sau proceduri) **begin** instrucțiuni; apeluri de subprograme; **end.**	*directive preprocesare* *includere fișiere bibliotecă header (antet)* **#include** <math.h> *definiții de constante;* **const** n=15; *definiții de tipuri de date;* **typdef float** sir[5]; *declarații de variabile;* **int** x,y; *declarații de subprograme* (funcții) ***void main()*** {instrucțiuni; apeluri de subprograme; }
Precizări: 1. Un program Pascal este un ansamblu de instrucțiuni, grupate în corpul programului principal și în subprograme, definite de programator — dacă acestea sunt necesare. 2. În orice program Pascal, pot fi folosite subprograme din unit-ul System, care nu trebuie declarat. 3. Corpul programului principal este delimitat prin **begin** și **end.** 4. Un bloc de instrucțiuni este delimitat prin **begin** și **end** 5. Fiecare instrucțiune se termină cu ; (punct virgulă).	*Precizări:* 1. Un program C/C++ este un ansamblu de instrucțiuni grupate în **funcții**. 2. Orice program C/C++ are cel puțin o **funcție** — funcția principală care se declară prin **void main ( )**. 3. Orice program C/C++ poate avea una sau mai multe funcții declarate de programator. 4. Un bloc de instrucțiuni este delimitat printr-o pereche de acolade { }. 5. Fiecare instrucțiune se termină cu ; (punct virgulă).

Exemplu:
Se citesc două numere întregi **a** și **b**; se afișează suma lor.

LIMBAJUL PASCAL	LIMBAJUL C/C++

program exemplu;
var a, b: **integer;**
begin {program principal}
write(' a = '); **readln**(a);
write(' b = '); **readln**(b);
writeln(' Suma a + b = ', a + b);
end.
	#include <iostream.h>
	int a,b;
	void main() // funcția principală
	{**cout**<<" a = "; **cin**>>a;
	cout<<" b = "; **cin**>>b;
	cout<<"Suma a + b = "<<a + b <<endl;
	}

1.3. Vocabularul limbajului de programare

Vocabularul oricărui limbaj de programare este format din: setul de caractere, identificatori, separatori și comentarii.

♦ **Setul de caractere**

Orice program este scris cu ajutorul următoarelor caractere:

— litere mari și mici ale alfabetului englez (A-Z, a-z), numite *caractere alfabetice*;

— cifrele sistemului de numerație zecimal, numite și *caractere numerice* (0-9);

— *caractere speciale*: +, -, *, /, =, &, [,], {, }, #, |, blank (spațiu) , _, ~, @.

Codificarea și reprezentarea informației alfanumerice folosește standardul ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).

♦ **Identificatori**

Un identificator reprezintă o succesiune de litere, cifre sau caracterul special „_"; primul caracter nu trebuie să fie cifră. Identificatorii pot avea orice lungime.

Exemple:

1. Identificatori: a, b1, cod_0, produs.
2. Succesiuni de caractere ce nu pot fi identificatori: 3*y* (primul caracter este o cifră), *ur+m* (conține un caracter special).

Orice identificator trebuie definit sau declarat într-o linie anterioară referirii sale.

Identificatorii desemnează constante, tipuri de date, variabile. Există un set de identificatori predefiniți, numiți ***cuvinte-cheie*** sau ***cuvinte rezervate*** (tabelul 2).

Tabelul 2

Cuvinte-cheie în limbajele de programare Pascal și C/C++

LIMBAJUL PASCAL	LIMBAJUL C/C++
and, or, while, for, do, repeat, array, mod, div, trunc, begin, end, type, procedure, function, nil.	while, void, for, do, struct, char, float, switch, NULL, include, const, floor, if, define.

♦ **Separatori**

Unitățile sintactice (ansambluri de caractere) sunt separate între ele fie prin unul sau mai multe spații libere (**blank**), fie prin sfârșitul de linie (caracterul **CR**), fie prin caracterul ; (punct virgulă) care se utilizează pentru separarea instrucțiunilor și a declarațiilor.

♦ **Comentarii**

În textul unui program, sunt necesare note explicative (comentarii) atașate unor secvențe de operații, declarări de tipuri de date/variabile, care nu au un rol activ în derularea programului. Acestea sunt delimitate în limbajul Pascal prin { ....}, iar în limbajul C/C++ sunt precedate de // .

TEMĂ

Se consideră următorii identificatori. Încercuiți litera corespunzătoare identificatorului corect definit. Justificați răspunsul!

a) n3 ; b) 4_nr ; c) stea ; d) p.adresa ; e) nr pare ; f) x + y ; g) mi#sol ; h) var_4.