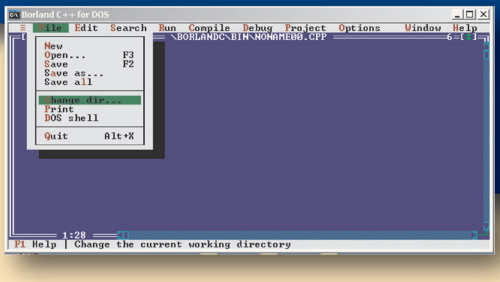
Ministerul Educaţiei şi Cercetării

Informatică

Profil real

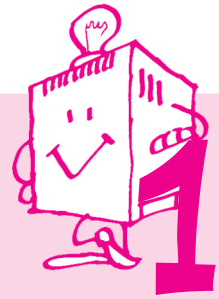


Mioara Gheorghe
(coordonator)

Constanţa Năstase
Monica Tătărâm

Manual pentru clasa a X-a

CORINT



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).

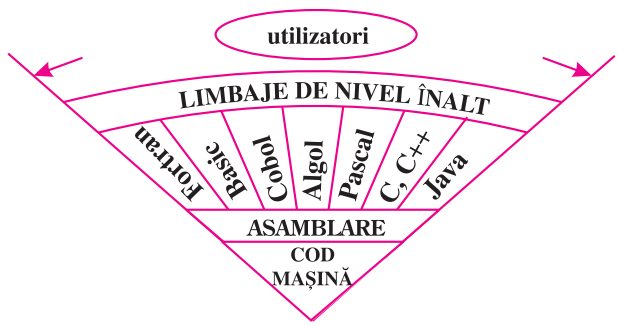


Figura 1. Evoluția limbajelor de programare

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.

1.4. Mediul limbajului de programare studiat

Prin codificarea algoritmului în limbajul de programare ales, obținem un program. Programul este transmis calculatorului prin introducerea textului de la tastatură (***editare***) și memorat într-un ***fișier sursă*** cu extensia **pas** pentru limbajul Pascal (exemplu: **program1.pas**) sau extensia **cpp** pentru limbajul C/C++ (exemplu: **program1.cpp**). Prelucrarea fișierelor sursă se face prin operațiile cunoscute: **New, Open, Save, Save As**.

„Traducerea" textului de program din limbajul de programare ales în limbajul intern al calculatorului se face de către ***compilator*** prin ***compilare***. Compilatorul realizează și verificarea sintactică a programului, semnalând erorile detectate. Corectarea erorilor se face prin ***editare***, de la tastatură, fiind urmată de o nouă compilare. Când textul pro-gramului nu mai conține nici o eroare, compilatorul generează un ***fișier obiect*** cu extensia **obj** (exemplu: **program1.obj**). Fișierul obiect este executat prin comanda **Run**.

Pentru a oferi programatorilor accesul la fișiere și la resursele de editare, compilare și execuție, au fost dezvoltate ***mediile de programare***.

Mediul de programare este o aplicație cu meniu interactiv care oferă o interfață accesibilă și prietenoasă (**user friendly**) pentru operațiile necesare dezvoltării, compilării, execuției și depanării programelor (FILE, EDIT, COMPILE, RUN, DEBUG).

♦ Descrierea unei sesiuni de lucru cu mediul de programare

Pas 1. Lansarea în execuție a mediului de programare

Mediul de programare se lansează fie de pe Desktop, prin activarea icon-ului corespunzător (shortcut), fie prin lansarea în execuție a fișierului executabil din folderul BIN numit **bp.exe** pentru limbajul Pascal și **bc.exe** pentru limbajul C/C++. *Exemplu:* C:\bp\bin**bp.exe** sau C:\borlandc\bin**bc.exe**.

Pas 2. Schimbarea folderului de lucru

În folderul de lucru se vor salva programele elaborate pe durata unei sesiuni de lucru. Pentru schimbare se alege **File → Change directory…** (figura 2).

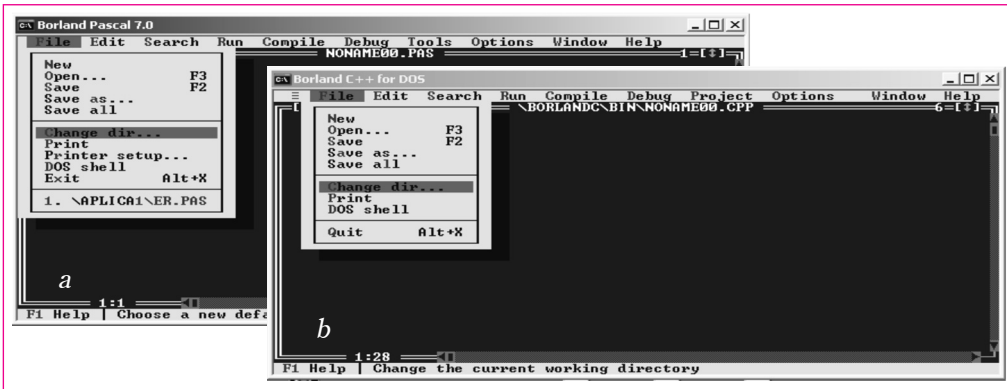


Figura 2. Schimbarea folder-ului de lucru: *a* — Pascal; *b* — C/C++

Pas 3. Accesul la fișiere

Crearea unui nou fișier sursă se realizează alegând calea **File → New**, iar deschiderea unui fișier sursă existent, alegând **File → Open**. În același mod, pot fi create/deschise și fișiere text cu date de intrare pentru testarea programelor.

Pas 4. Salvarea

Salvarea fișierului în folderul ales se realizează prin **File → Save** (sau apăsând tasta funcțională **F2**). Dacă se dorește redenumirea fișierului, se alege **File → Save as** …

Pas 5. Editarea programului

Editarea liniilor de program (figura 3) se face de la tastatură. Nu uitați să salvați permanent cu tasta **F2**!

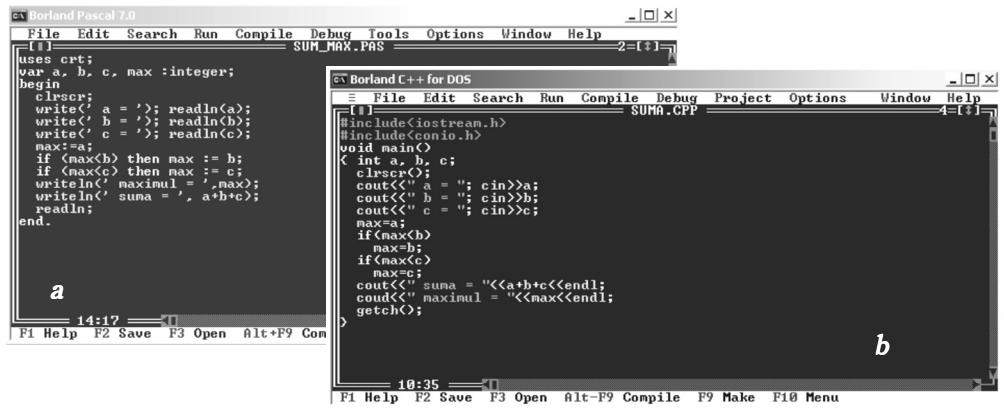


Figura 3. Editarea liniilor de program: *a* — Pascal; *b* — C/C++

Pas 6. Compilarea

Compilarea liniilor de program (figura 4) se realizează alegând calea **Compile → Compile** sau prin activarea simultană a tastelor **ALT+F9**. Nu uitați să salvați permanent cu tasta **F2**!

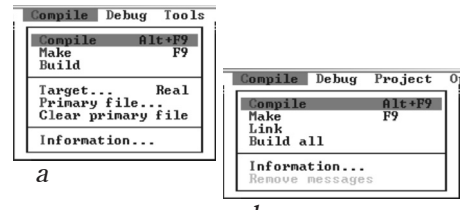


Figura 4. Compilarea: *a* — Pascal; *b* — C/C++

Pas 7. Lansarea în execuție

Lansarea în execuție a programului (figura 5) se realizează alegând calea **Run → Run** sau prin activarea simultană a tastelor **CTRL+F9**.

Dacă, după execuție, se observă că rezultatele obținute nu sunt

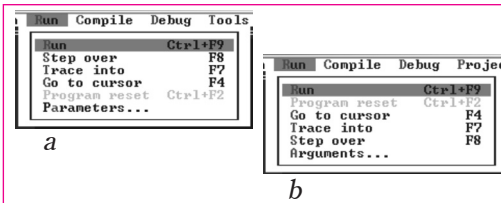


Figura 5. Lansarea în execuție:
a — Pascal; *b* — C/C++

corecte, se verifică algoritmul de rezolvare a problemei; se reiau pașii 5-7 până când programul furnizează date de ieșire corecte.

Pentru a înțelege cum sunt executate liniile de program (instrucțiuni, operații, apeluri de subprograme etc.), se poate executa programul **pas cu pas** din opțiunea **Run → Trace into (F7)** sau **Run → Step over (F8)**.

În tratarea erorilor de concepție a programului, este utilă opțiunea din meniul sistem **Debug → Add Watches** (în ferestrele de dialog sunt trecuți identificatorii variabilelor ale căror modificări interesează la rularea programului).

Pas 8. Închiderea și părăsirea mediului de programare

Închiderea și părăsirea mediului de programare se realizează alegând combinația de taste **ALT + X** sau: **File → Exit** pentru Pascal și **File → Quit** pentru C/C++.

TEME

Deschideți o sesiune de lucru în mediul de programare al limbajului studiat și rezolvați următoarele cerințe:

1. Editați un fișier sursă cu textul de program de la pasul 5 și executați operațiile descrise la pașii 6 și 7.

2. Inspectați meniul mediului de programare și descrieți succint opțiunile și tastele funcționale.

3. Explicați semnificația următoarelor operații: editare, compilare, execuție.

4. Creați și salvați fișierul text **date.in** cu date personale (note, agendă).

2. DESCRIEREA ȘI PRELUCRAREA DATELOR

2.1. Tipuri standard de date

Datele prelucrate de un program corespund datelor identificate în etapa de analiză a problemei. În schema din figura 6, este prezentată clasificarea datelor după natura și stabilitatea lor.

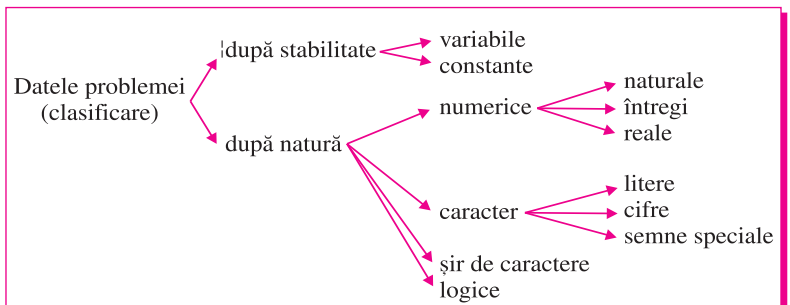


Figura 6. Clasificarea datelor unei probleme

Un tip de dată este caracterizat prin mulțimea de valori și dimensiunea zonei de memorie alocată (numărul de octeți).

Pentru început, vor fi studiate doar câteva tipuri de date necesare unui programator începător.

Din punct de vedere al complexității, există două categorii de date (tabelul 3):

- **date simple** (sau elementare);
- **date structurate** (sau compuse) — se obțin din tipurile simple prin:

a) compunerea, în memoria internă, a unui număr finit de elemente de același tip (tablouri de date);

b) compunerea, în memoria externă, a unui număr nelimitat de componente (fișiere).

Tabelul 3

Tipuri standard de date

Tipuri de date	LIMBAJUL PASCAL		LIMBAJUL C/C++	
	Date	**Număr de octeți**	**Date**	**Număr de octeți**
Tipuri simple	• întregi — **integer**	2	• întregi — **int**	2
	— **shortint**	1	— **shortint**	2
	— **longint**	4	— **unsigned int**	2
	— **byte**	1	— **long**	4
	— **word**	2	— **unsigned long**	4
	• reale — **real**	6	• reale — **float**	4
	— **single**	4	— **double**	8
	— **double**	8	— **long double**	10
	— **extended**	10		
	• caracter — **char**	1	• caracter — **char**	1
	• logice — **boolean**	1	*Precizare:* Orice valoare de tip întreg diferită de zero are semnificația de valoare logică **adevărat**; zero semnifică valoarea logică **fals**.	
Tipuri structurate	• tablouri de date (vectori, șiruri) • fișiere text — **text**		• tablouri de date (vectori, șiruri) • fișiere text — **fstream**	

Dimensiunea zonei de memorie limitează domeniul de valori. De exemplu: pentru întregi cu semn avem [–32786, 32767], iar pentru întregi fără semn, [0, 65535].

2.2. Constante, variabile, expresii

2.2.1. Constante și variabile

Valorile atribuite ***constantelor*** nu se pot modifica prin operațiile sau instrucțiunile din program. Tipurile de constante și definirea acestora sunt prezentate în tabelul 4.

Zonele de memorie al căror conținut poate fi modificat în timpul execuției programului se numesc ***variabile***. O variabilă poate fi declarată pentru orice tip standard de date.

O variabilă poate primi valori prin citire direct de la tastatură.

O variabilă poate fi afișată prin scriere pe ecranul calculatorului.

Tabelul 4

Tipuri de constante/definirea constantelor și a variabilelor

LIMBAJUL PASCAL	LIMBAJUL C/C++
Tipuri de constante	
• **întregi** — numere întregi *pozitive* care nu pot depăși valoarea 32767, valoare identificată de compilator sub numele de MAXINT; valorile întregi pot fi numere: — *zecimale* (baza 10); Exemple: 43, 187, 2001. — *hexazecimale* (baza 16), care sunt precedate de caracterul special $. Exemplu: $2F4 sau $2F4 → $2F4_{(16)}$. • **reale** — numere reale cu valori absolute cuprinse între 5.9×10^{-39} și 3.4×10^{38}; atât partea întreagă, cât și cea zecimală trebuie să conțină cel puțin o cifră, chiar dacă aceasta este 0. • **caracter** — orice caracter scris între apostrofuri. Exemplu: 'A' • **șiruri de caractere** — o succesiune de caractere, delimitată tot prin apostrofuri. • **constante definite prin cuvinte-cheie** Exemple: true/false, MAXINT.	• **întregi** — numere întregi *pozitive* cu valori între 0 și 4.294.967.295 și care sunt de trei tipuri: — *zecimale* (baza 10); Exemple: 43, 152, 7564. — octale (baza 8), care sunt precedate de un zero nesemnificativ; Exemplu: 0354 → $345_{(8)}$. — *hexazecimale* (baza 16), care sunt precedate de 0x sau 0X. Exemplu: 0x2F4 sau 0X2F4 sau 0x2f4 - → $2F4_{(16)}$. • **reale** — orice valoare reală Exemple: 32.45, 0.5. • **caracter** — orice caracter scris între apostrofuri. Exemplu: 'A' • **șiruri de caractere** — o succesiune de caractere, delimitată tot prin ghilimele. • **constante definite prin cuvinte-cheie** Exemplu: NULL
Definirea constantelor și a variabilelor	
const id_constanta = valoare; **var** id_variabilă: tip standard; Exemplu: **const** pi = 3.14; **const** unu = '1'; **var** s : **integer**; p : **real**;	**const** [tip] id_constantă = valoare; tip standard id_variabilă; Exemplu: **const** pi = 3.14; **const** unu = '1'; **int** s; **float** p;

TEME

Stabiliți tipul constantelor din tabelul de mai jos.

LIMBAJUL PASCAL	132	$5BF	–2.34	512E+23	'Succesiune'	MAXLONGINT	'A'
LIMBAJUL C/C++	132	Ox5BF	–2.34	512E+23	"Succesiune"	NULL	'A'

2.2.2. Expresii

O ***expresie*** este formată din unul sau mai mulți operanzi (constante, variabile, funcții), legați prin opratori. O expresie are o **valoare** și un **tip**.

În evaluarea unei expresii, ordinea efectuării operațiilor respectă **regulile de prioritate** ale operatorilor. Pentru a schimba o prioritate se utilizează paranteze rotunde. În tabelele 5 și 6, sunt prezentate principalele categorii de operatori și prioritatea acestora, corespunzător fiecărui limbaj de programare.

Tabelul 5

Tipuri de operatori

LIMBAJUL PASCAL	LIMBAJUL C/C++
1. Operatori aritmetici: — unari: + , - — binari multiplicativi: *, /, **div** (cât), **mod** (rest) — binari aditivi: +, - **2.** Operatori relaționali: <, <=, >, >= **3.** Operatori de egalitate: = (egal), <> (diferit) **4.** Operatori logici: **not** (negare logică, operator unar), **and** (*și* logic), **or** (*sau* logic), **xor** (*sau exclusiv*)	**1.** Operatori aritmetici: — unari: + , - — binari multiplicativi: *, /, **%** (rest) — binari aditivi: +, - **2.** Operatori relaționali: <, <=, >, >= **3.** Operatori de egalitate: = = (egal), **!**= (diferit) **4.** Operatori logici: **!** (negare logică, operator unar), **&&** (*și* logic), ‖ (*sau* logic) **5.** Operatori logici pe biți: ~ (complement față de 1, operator unar), << >> (deplasare pe biți la stânga/la dreapta), **&** (*și* pe biți), ^ (*xor* – sau exclusiv pe biți), ¦ (*sau* pe biți)
not(0) → 1 not(1) →0	~ 0 → 1 ~ 1 → 0
and 0 1 0 0 0 1 0 1	**&** 0 1 0 0 0 1 0 1
or 0 1 0 0 1 1 1 1	¦ 0 1 0 0 1 1 1 1
xor 0 1 0 0 1 1 1 0	^ 0 1 0 0 1 1 1 0
5. Operatori de atribuire: **: =** id_v : = valoare/expresie	**6.** Operatori de atribuire: id_v = valoare/expresie Pentru operația de atribuire se pot utiliza și următoarele asocieri de operatori: *=, /=, %=, +=, -=, <<=, >>=, &=, = , ^=
6. Operatorii de incrementare/decrementare: **inc**(id_v) ⇔ id_v : = id_v + 1 **dec**(id_v) ⇔ id_v : = id_v – 1 unde id_v este o variabilă de tip întreg.	**7.** Operatorii de incrementare/decrementare: postfixat: id_v++ / id_v- - prefixat : ++id_v / -- id_v Dacă operatorul este postfixat, atunci variabila este incrementată/decrementată după evaluarea expresiei din care face parte; dacă operatorul este prefixat, atunci variabila este incrementată/decrementată înainte de evaluarea expresiei.
7. Operatorul **sizeof(expresie)/ sizeof(tip)** returnează numărul de octeți alocați pentru memorarea unei expresii sau a unui tip de dată.	**8.** Operatorul **sizeof(expresie)/ sizeof(tip)** returnează numărul de octeți alocați pentru memorarea unei expresii sau a unui tip de dată. **9.** Operatorul condițional: e1 ? e2 : e3 unde e1, e2, e3 sunt expresii. Dacă e1 are valoare nenulă, atunci se prelucrează e2, altfel, se prelucrează e3. **10.** Operatorul de conversie explicită **(tip)operand** convertește valoarea operandului la tipul indicat.

Tabelul 6

Operatori — reguli de prioritate și evaluare

LIMBAJUL PASCAL			LIMBAJUL C/C++		
Prioritate	**Operator**	**Evaluare**	**Prioritate**	**Operator**	**Evaluare**
1	**( )**	s→d	1	**( )**	s→d
2	**not + - sizeof()**	d→s	2	**! ~ + - ++ — sizeof()**	d→s
3	*** / div mod**	s→d	3	*** / %**	s→d
4	**+ -**	s→d	4	**+ -**	s→d
5	**< <= > >=**	s→d	5	**< <= > >=**	s→d
6	**= <>**	s→d	6	(egal) **==** **!=** (diferit)	s→d
7	**and** (*și* logic)	s→d	7	**&** (și pe biți)	s→d
8	**xor**	s→d	8	**^** (**XOR** pe biți)	s→d
9	**or** (*sau* logic)	s→d	9	**|** (**OR** pe biți)	s→d
10	**:=** (atribuire)	s→d	10	**&&** (*și* logic)	s→d
			11	**||** (*sau* logic)	s→d
			12	**=** (atribuire)	d→s

Exemplu:
Avem următoarele expresii:

LIMBAJUL PASCAL	LIMBAJUL C/C++
expresii numerice — determinantul ecuației de gradul II	
D : = b * b – 4 * a * c	D = b * b – 4 * a * c
expresii logice — x ∈ [a, b]	
(x > = a) and (x < = b)	(x > = a) && (x < = b)
	expresii la nivel de bit
	a = 3 b = 2 c = a & b = 2 a → 0 1 1 b → 0 1 0 a & b → 0 1 0 → 2

TEME

1. Scrieți, în limbajul de programare studiat, următoarele expresii:
E1 = „x este număr par și y nu se divide la 3, 5 și 7“
E2 = „x este mai mic sau cel mult egal cu y și y este multiplu de 11 și 9“.

2. Evaluați următoarele expresii pentru a = 5, b = 2, c = 3:
E1 = a + b/2 + c * a + b
E2 = -c + b * a + (c * b / a + b + c) / (a * b).

3. Fie numere reale **a**, **b**, **c**, **d**, și **x**, unde **a** < **b** și **c** < **d**. Identificați expresia corectă pentru ca **x** ∈ [**a**, **b**) sau **x** ∈ (**c**, **d**].

LIMBAJUL PASCAL	LIMBAJUL C/C++
a) (x>=a or x <b) and (x>c or x<=d) b) ((x>=a) and (x<b)) or ((x>c) and (x<=d)) c) (x>=a or x <b) or (x>c or x<=d) d) ((x>=a) and (x<=b)) or ((x>=c) and (x<=d))	a) (x>=a|| x <b) && (x>d|| x<=d) b) ((x>=a) && (x<b)|| ((x>c) && (x<=d)) c) (x>=a || x <b)|| (x>d|| x<=d) d) ((x>=a) && (x<=b)|| ((x>=c) && (x<=d))

2.3. Citirea și scrierea datelor

Operațiile de citire a datelor de intrare și de scriere a datelor de ieșire se pot realiza prin două căi:

1. de la tastatură/pe ecran; simbolizate prin (1) pe figura 7;
2. din fișiere text/în fișiere text; simbolizate prin (2) pe figura 7.

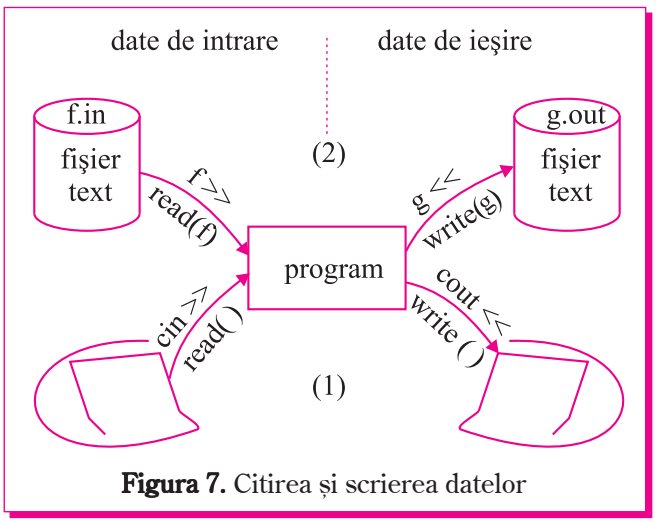


Figura 7. Citirea și scrierea datelor

Pentru fiecare tip de operație de citire sau scriere, există comenzi specifice implementate în bibliotecile limbajelor de programare.

Biblioteca limbajului Pascal se află în unit-ul **System**.

Limbajul C/C++ are mai multe astfel de biblioteci; în manual, vom folosi **iostream.h** pentru operațiile standard și **fstream.h** pentru fișiere.

2.3.1. Operații cu tastatura și ecranul

Operațiile de citire a datelor de intrare de la tastatură și de scriere (afișare) a celor de ieșire pe ecran sunt prezentate în tabelul 7. Valorile afișate pot fi cele reținute în zonele de memorie sau cele rezultate din evaluarea unor expresii.

Tabelul 7

Operații cu dispozitivele standard

LIMBAJUL PASCAL	LIMBAJUL C/C++
CITIREA DE LA INTRAREA STANDARD (TASTATURĂ) se face cu:	
procedurile **read** și **readln** ***read***(id_v1[, idv2,.., id_vn]); ***readln***(id_v1[, idv2,.., id_vn]); unde id_v1, id_v2, ..., id_vn sunt identificatori de variabile elementare de orice tip, cu excepția celor logice. Procedura **readln** are același rol cu cel al procedurii **read**, cu diferența că, la terminarea citirii, cursorul este poziționat pe linia următoare a ecranului.	funcția **cin** #include <iostream.h>; ***cin***>>id_v1[>>id_v2>>...>>id_vn]; unde id_v1, id_v2, ..., id_vn sunt identificatori de variabile elementare de orice tip.
SCRIEREA (AFIȘAREA) LA IEȘIREA STANDARD (ECRAN) se face cu:	
procedurile **write** și **writeln** **write** (id1[, id2,.., idn]); **writeln** (id1[, id2,.., idn]); unde id1, id2, ..., idn sunt identificatori de variabile elementare de orice tip sau de constante. La terminarea scrierii (afișării), cu procedura **writeln**, cursorul este adus pe prima poziție a rândului următor.	funcția **cout** #include<iostream.h>; **cout<<**id_v1[<<id_v2<<..<<id_vn]; unde id_v1, id_v2, ..., id_vn sunt identificatori de variabile elementare de orice tip sau de constante.

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

LIMBAJUL PASCAL	LIMBAJUL C/C++

program produs;
var a, b, p: integer;
begin
write ('a='); **readln** (a);
write ('b='); **readln** (b);
p:= a * b;
writeln ('a*b=', p);
end.
	#include<iostream.h>;
	void main ( )
	{
	int a, b, p;
	cout<<"a="; cin>>a;
	cout<<"b="; cin>>b;
	p=a*b;
	cout<<"a*b="<<p;
	}

2.3.2. Operații cu fișiere text

Datele cu care lucrează un program pot fi memorate în fișiere text. Astfel se asigură păstrarea datelor și reducerea timpului de operare la testarea/utilizarea programului.

Un ***fișier text*** este format dintr-o succesiune de caractere ASCII scrise pe una sau mai multe linii (rânduri), pe memorie externă (disc). Sfârșitul fișierului este marcat

de o „etichetă“ specială: EOF (**E**nd **O**f **F**ile). Prelucrarea unui fișier se face prin mai multe operații: *deschidere* (pentru citire sau scriere), *citire*, *scriere*, *închidere* (tabelul 8). La nivel de program, fișierul este recunoscut printr-o variabilă asociată, numită *identificator de fișier* (**id_f**).

Tabelul 8

Operații cu fișiere de text

LIMBAJUL PASCAL	LIMBAJUL C/C++
CITIREA DATELOR DIN FIȘIERE TEXT are următoarele etape:	
• definirea identificatorului de fișier ***var*** id_f:***text***; • asocierea dintre fișierul fizic și identificator: ***assign***(id_f,'nume_f'); • deschiderea fișierului pentru citire: ***reset***(id_f); • citirea variabilelor din fișier se face cu ajutorul procedurilor: ***read*** și ***readln***. ***read***(id_f,id_v1[, idv2,.., id_vn]); ***readln***(id_f,id_v1[, idv2,.., id_vn]); unde id_v1, id_v2, ..., id_vn sunt identificatori de variabile elementare de orice tip, cu excepția celor logice. • închiderea fișierului: ***close***(id_f).	#include<fstream.h> • implementarea fișerelor de tip text: **fstream** id_f([cale\\]nume_f,ios::in); sau **ifstream** id_f("nume_f"); Prin implementare, fișierul este deschis pentru citire. • citirea se face astfel: ***id_f***>>id_v1[>>id_v2>>...>>id_vn]; unde id_v1, id_v2, ..., id_vn sunt identificatori de variabile elementare de orice tip. • închiderea fișierului : id_f.**close**();
SCRIEREA ÎN FIȘIERE TEXT are următoarele etape:	
• definirea identificatorului de fișier ***var*** id_f:***text***; • asocierea dintre fișierul fizic și identificator: ***assign***(id_f,'nume_f'); • deschiderea fișierului pentru scriere: ***rewrite***(id_f); • scrierea variabilelor în fișier se face cu ajutorul procedurilor **write** și **writeln**: ***write***(id_f, id1[, id2,.., idn]); ***writeln***(id_f, id1[, id2,.., idn]); unde id1, id2, ..., idn sunt identificatori de variabile elementare de orice tip sau de constante. • închiderea fișierului: ***close***(id_f);	#include<fstream.h> • implementarea fișerelor de tip text: **fstream** id_f([cale\\]nume_f,ios::out); sau **ofstream** id_f([cale\\]"nume_f"); Prin implementare, fișierul este deschis pentru scriere. • scrierea în fișier se face astfel: ***id_f***<<id_v1[<<id_v2<<..<<id_vn]; unde id_v1, id_v2, .., id_vn sunt identificatori de variabile elementare de orice tip sau de constante. • închiderea fișierului: id_f.**close**();

Un fișier text poate fi creat direct din mediul de programare **File → New**, salvat cu extensia dorită (exemplu: **date.in**) și folosit ("citit") în program.

Exemplu:
Din fișierul **date.in** (creat în directorul curent), se citesc variabilele reale **a**, **b** și **c**, separate prin câte un spațiu.

În fișierul **date.out** se vor afișa valorile citite, pe prima linie, cu câte un spațiu între ele, iar pe următoarele linii, suma și produsul lor.

date.in	**date.out**
3 4 5	3 4 5 12 60

LIMBAJUL PASCAL	LIMBAJUL C/C++

var f , g : text;
a, b, c , s, p : real;
begin
assign(f , 'date.in'); reset(f);
assign(g , 'date.out'); rewrite(g);
read(f , a , b , c);
s: = a + b + c ; prod : = a*b*c;
writeln(g, a, ' ',b,' ',c); writeln(g, 'Suma = ',s);
writeln(g, ' Produsul = ',p);
close(f); close(g);
end.
	#include<fstream.h>
	ifstream f("date.in");
	ofstream g("date.out");
	void main()
	{float a, b, c, s, p;
	f>>a>>b>>c;
	s=a+b+c; p=a*b*c;
	g<<a<<" "<<b<<" "<<c<<endl;
	g<<"Suma = "<<s<<endl;
	g<<"Produsul= "<<p<<endl;
	f.close(); g.close();
	}

TEME

1. Analizați exemplul corespunzător limbajului de programare studiat și explicați operațiile de lucru cu fișiere.

2. Modificați secvența din exemplu astfel încât introducerea datelor să se facă de la tastatură.

3. Modificați secvența din exemplu astfel încât scrierea datelor să se facă pe ecran.

4. Creați fișierul **date.in** și testați programul. Pentru verificarea rezultatelor, deschideți fișierul **date.out**.

2.4. Funcții matematice uzuale

Limbajele de programare au biblioteci în care se află subprograme pentru determinarea valorilor celor mai frecvente funcții matematice. Biblioteca limbajului Pascal se află în unit-ul **System**, iar cea a limbajului C/C++, în fișierul header <**math.h**>.

Pentru funcțiile matematice prezentate în manual (tabelul 9), sunt definite tipurile de date ale domeniilor de definiție și tipurile de date ale mulțimilor de valori.

Tabelul 9

Funcții matematice

LIMBAJUL PASCAL		LIMBAJUL C/C++	
		#include<math.h>	
abs : R → R_+	**abs**(x) — modulul lui x	abs : Z → N	**abs**(x) — modulul lui x
abs : Z → N		(vezi labs, fabs)	
sqr : R → R_+	**sqr**(x) — x la puterea a doua	pow : R → R_+	**pow**(x,2) — x la puterea a doua
sqrt : R_+ → R_+	**sqrt**(x) — radical din x	sqrt : R_+ → R_+	**sqrt**(x) — radical din x
sin : R → [-1,1]	**sin**(x), **cos**(x), **arctan**(x) —	sin : R → [-1,1]	**sin**(x), **cos**(x), **arctan**(x) —
cos : R → [-1,1]	funcții trigonometrice	cos : R → [-1,1]	funcții trigonometrice
arctan : R → R		atan : R → R	
ln : R_+ → R	**ln**(x) — logaritm natural	log : R_+ → R	**log**(x) — logaritm natural (ln x)
		log10 : R_+ → R	**log10**(x) — logaritm în baza 10 (lg x)
exp : R → R_+	**exp**(x) — e^x (e = 2,71 constanta Euler)	exp : R → R_+	**exp**(x) — e^x (e = 2,71 constanta Euler)
int : R → Z	**int**(x) — partea întreagă din x	ceil : R → Z	**ceil**(x) — rotunjește la cel mai apropiat întreg mai mare decât x
trunc : R → Z	**trunc**(x) — rotunjește la cel mai apropiat întreg mai mic decât x		
round : R → Z	**round**(x) — rotunjește la cel mai apropiat întreg mai mare decât x	floor : R → Z	**floor**(x) — rotunjește la cel mai apropiat întreg mai mic decât x
frac : R → (-1,1)	**frac**(x) — returnează partea fracționară a lui x	pow : R × R → R	**pow**(x,y) — calculează x^y

TEME

1. Stabiliți ce afișează fiecare dintre instrucțiunile din tabelul alăturat.

LIMBAJUL PASCAL	LIMBAJUL C/C++
writeln(round (5.62)) ;	cout<<ceil(5.62)<<endl;
writeln(trunc(4.32)) ;	cout<<floor(4.32)<<endl;
writeln(round(-7.28));	cout<<ceil(-7.28)<<endl;
writeln(trunc(-4.33));	cout<<floor(-4.33);
writeln(abs(-14));	cout<<abs(-14)<<endl;

2. Transformați fiecare instrucțiune într-o secvență care să cuprindă declararea, citirea și scrierea datelor, ca în exemplul din tabelul alăturat.

LIMBAJUL PASCAL	LIMBAJUL C/C++
var x: **real;**	**float** x;
read(x);	**cin**>>x;
write(**round** (x));	**cout**<<**ceil**(x);

CUPRINS