MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII

INFORMATICĂ

Manual pentru clasa a 8-a

ANATOL GREMALSCHI • IURIE MOCANU • ION SPINEI
LUDMILA GREMALSCHI

Știinţa, 2020

CZU 004(075.3)
I-52

Manualul şcolar a fost realizat în conformitate cu prevederile Curriculumului la disciplină, aprobat prin Ordinul Ministerului Educaţiei, Culturii şi Cercetării nr. 906 din 17 iulie 2019. Manualul a fost aprobat prin Ordinul Ministerului Educaţiei, Culturii şi Cercetării nr. 1048 din 28.09.2020, urmare a evaluării calităţii metodico-ştiinţifice.

Editat din sursele financiare bugetare.

Comisia de evaluare:
Coordonator: *Natalia Schiţco*, grad didactic superior, LT „Socrate", Chişinău;
Svetlana Brînză, grad didactic superior, LT „N.M. Spătaru", Chişinău;
Viorica Juc, grad didactic superior, LT „Ion Creangă", Chişinău;
Olga Turchin, grad didactic unu, LT „V. Suhomlinschi", Edineţ;
Diana Bagrin, grad didactic unu, LT „Da Vinci", Chişinău

Denumirea instituției de învățământ _____

Manualul a fost folosit: _____

Anul de folosire	Numele, prenumele elevului	Anul de studii	Aspectul manualului	
			la primire	la returnare
1				
2				
3				
4				
5				

Dirigintele verifică dacă numele, prenumele elevului sunt scrise corect.

Elevii nu vor face niciun fel de însemnări în manual.

Aspectul manualului (la primire şi la returnare) se va aprecia cu unul dintre următorii termeni: *nou, bun, satisfăcător, nesatisfăcător.*

Responsabil de ediţie: Larisa Dohotaru
Redactor: Mariana Belenciuc
Corector: Maria Cornesco
Redactor tehnic: Nina Duduciuc
Machetare computerizată: Vitalie Ichim
Copertă: Romeo Şveţ

ÎNTREPRINDEREA
EDITORIAL-POLIGRAFICĂ
ȘTIINȚA

str. Academiei, nr. 3; MD-2028, Chişinău, Republica Moldova
tel.: (+373 22) 73-96-16 (anticamera); (+373 22) 73-99-94 (marketing)
(+373 22) 73-99-83 (depozit); fax: (+373 22) 73-96-27
e-mail: prini_stiinta@yahoo.com; www.editurastiinta.md

Descrierea CIP a Camerei Naţionale a Cărţii
Informatică: Manual pentru clasa a 8-a / Anatol Gremalschi, Iurie Mocanu, Ion Spinei, Ludmila Gremalschi; Comisia de evaluare: Natalia Schiţco (coordonator) [et al.]; Ministerul Educaţiei, Culturii şi Cercetării. – Ch.: Î.E.P. *Ştiinţa*, 2020 (Tipogr. „Balacron"). – 144 p.: fig., tab.
Proprietate a Min. Educaţiei, Culturii şi Cercet.

ISBN 978-9975-85-235-7

Imprimare la Tipografia **BALACRON** SRL
str. Calea Ieşilor, 10, MD-2069, Chişinău, Republica Moldova
Comanda nr. 828

CUPRINS

Dragi prieteni,

Informatica ne schimbă viaţa, uneori într-un mod cu totul neaşteptat chiar şi pentru cei iniţiaţi în domeniu. Pentru a stăpâni schimbarea, pentru a fi competitivi şi a beneficia de roadele acestei ştiinţe, trebuie să cunoaştem şi să putem aplica întregul arsenal de metode şi tehnici informaţionale.

S-a stabilit că performanţele calculatoarelor moderne – viteza de operare, capacitatea memoriei interne, componenţa şi caracteristicile unităţilor periferice – se dublează practic în fiecare an. Acelaşi progres spectaculos se atestă şi în domeniul comunicaţiilor, fapt ce permite conectarea tuturor calculatoarelor într-o reţea globală, care conţine cantităţi enorme de informaţii. Aceste informaţii sunt produse de oameni şi pentru oameni. În vederea utilizării eficiente a informaţiilor acumulate şi contribuirii la crearea de noi cunoştinţe, fiecare om trebuie să-şi formeze şi să-şi dezvolte cultura informaţională şi gândirea algoritmică. Această cultură presupune cunoaşterea profundă a concepţiilor de bază ale informaticii şi capacitatea oricărei persoane de a elabora algoritmi pentru soluţionarea problemelor pe care le întâmpină în activitatea cotidiană.

Dacă la începutul secolului trecut, pentru a face faţă provocărilor timpului, de la fiecare membru al societăţii se cerea ştiinţă de carte, astăzi, la început de mileniu, de la fiecare om se cere, de asemenea, tot mai insistent "ştiinţă de calculator". Nu în zadar în mai multe limbi ale lumii a apărut termenul "licenţă (permis) de conducere a calculatorului". Conform datelor de ultimă oră, fără acest permis, majoritatea profesiilor moderne, inclusiv cele tradiţionale, devin inaccesibile.

Manualul de faţă are drept scop însuşirea de către elevi a cunoştinţelor necesare pentru prelucrarea automată a documentelor cu ajutorul editoarelor de texte şi formarea gândirii algoritmice.

Fiind cele mai răspândite programe de calculator, aplicaţiile de procesare a textelor sunt utilizate pentru: introducerea datelor, inserarea obiectelor multimedia, prezentarea informaţiilor într-o formă atractivă. Ca şi în cazul studierii altor programe de calculator, aplicaţiile de procesare a textelor reprezintă un instrument puternic, care ne ajută să cunoaştem metodele de păstrare şi prelucrare a informaţiei, de organizare a datelor în tabele, de reprezentare a acestora în formă de diagrame. Cunoştinţele teoretice şi practice din domeniul aplicaţiilor de procesare a textelor vor fi de un real folos nu numai după absolvirea gimnaziului, dar şi în procesul studierii altor discipline şcolare, cum ar fi matematica, fizica, chimia, geografia, istoria etc.

Formarea gândirii algoritmice presupune înţelegerea noţiunilor de executant şi algoritm, cunoaşterea formelor de reprezentare şi a proprietăţilor algoritmilor. Materiile teoretice şi practice, incluse în manual, vă vor ajuta să faceţi primii paşi în lumea algoritmilor, să elaboraţi propriile programe de calculator.

În scopul dezvoltării vocaţiei fiecăruia dintre elevi, manualul conţine şi materii opţionale: implementarea algoritmilor în medii textuale de programare şi prelucrarea imaginilor. Studierea acestor materii se va face, în principal, prin învăţarea independentă, asistată de tutoriale online, prin diverse activităţi practice, prin elaborarea de proiecte originale, strâns legate de viaţa şcolară, de satele şi oraşele în care locuiţi.

Vă dorim succese!

Autorii

Capitolul 1

EDITAREA TEXTELOR

1.1. Aplicaţia *Word*

Termeni-cheie:
- document
- structură a documentului
- fereastră de document
- panglică de instrumente

Aplicația **Word** operează cu fişiere speciale, numite *documente*.

***Documentul* reprezintă un obiect complex, format din obiecte mai simple: texte, tabele, imagini, secvențe sonore şi secvențe video procesate unitar.**

Structura ierarhică a documentului este prezentată în *figura 1.1*.

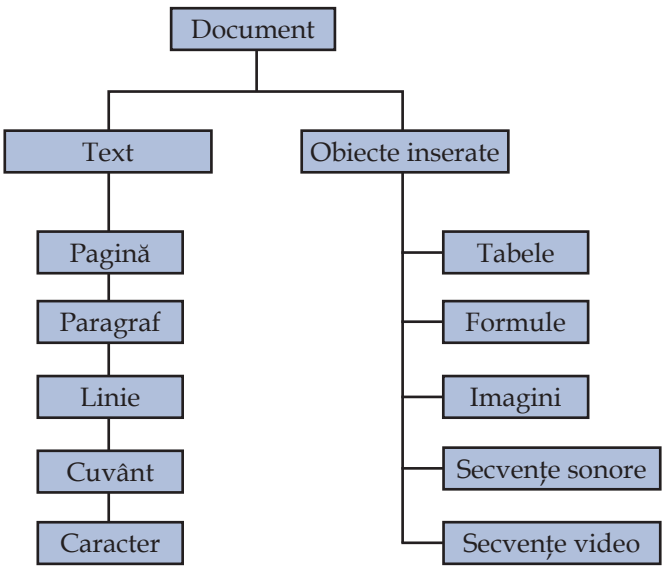


Fig. 1.1. Structura ierarhică a documentului

Fiecare obiect din componența unui document are anumite **proprietăți**. Aceste proprietăți pot fi modificate cu ajutorul instrumentelor aplicației **Word**. De exemplu, textul se caracterizează prin tipul şi dimensiunile caracterelor utilizate,

tabelul – prin numărul de linii şi coloane, desenul – prin dimensiuni şi amplasarea în pagină etc. Cu ajutorul instrumentelor respective, utilizatorul poate schimba aspectul şi dimensiunile caracterelor selectate, insera sau şterge linii şi coloane din tabele, mări sau micşora desenele. Fereastra aplicației **Word** este prezentată în *figura 1.2*.

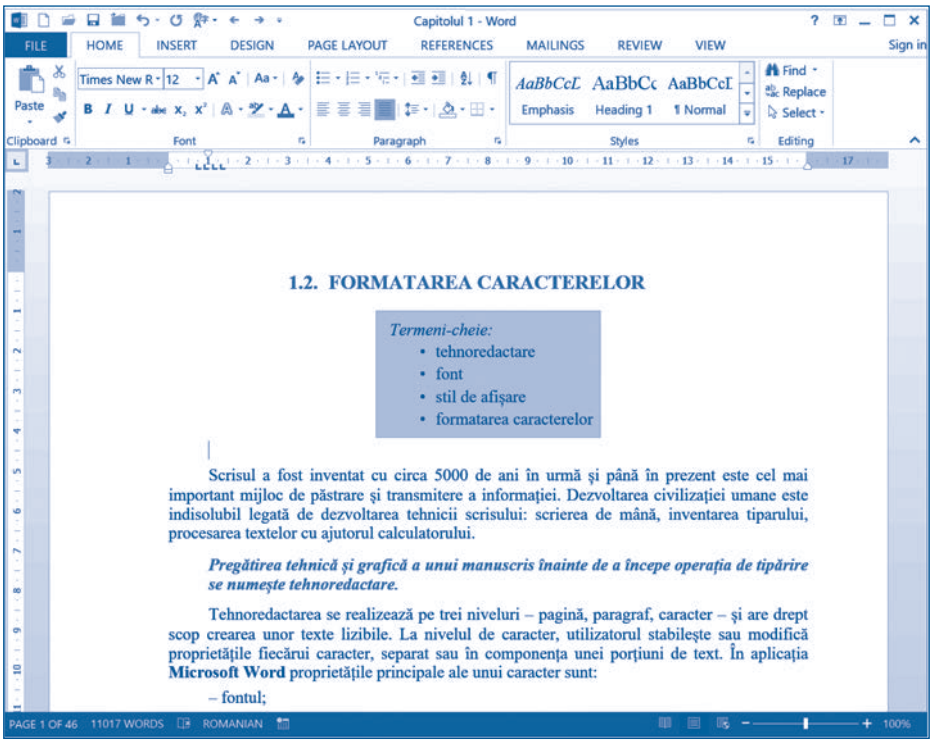


Fig. 1.2. Fereastra aplicației **Word**

Această fereastră include următoarele elemente:
– bara de titlu;
– panglica de meniuri şi instrumente standard;
– aria de lucru;
– linia de stare;
– zona-text;
– bara verticală de defilare;
– bara orizontală de defilare;
– rigla verticală;
– rigla orizontală.

În funcție de versiunea aplicației, fişierele create de aplicația **Word** au extensia **.doc** (document) sau **.docx**. Citirea şi salvarea documentelor se realizează cu ajutorul comenzilor **New**, **Open**, **Close** (Închide), **Save**, **Save As** din meniul **File**. Accesul rapid la unele comenzi se asigură prin acționarea butoanelor respective din panglica de instrumente. Destinația butoanelor poate fi aflată poziționând cursorul pe fiecare dintre ele.

Ca şi în cazul aplicației **Notepad**, textul unui document poate fi introdus de la tastatură. Operațiile de editare a textului se realizează cu ajutorul comenzilor **Cut**, **Copy**, **Paste**, **Clear** (Şterge), **Find**, **Replace** (Înlocuieşte) din meniul **Home** sau al butoanelor respective din panglica de instrumente.

Spre deosebire de programul **Notepad**, aplicația **Word** trece automat în linia următoare cuvintele care nu încap în linia curentă. Tasta <*Enter*> se acționează numai pentru a marca sfârşitul unui paragraf. În mod similar, când se termină pagina curentă, aplicația trece automat la o pagină nouă.

Întrebări şi exerciții

❶ Ce obiecte poate conține un document? Dați exemple.

❷ Prin ce se caracterizează obiectele din componența unui document? Dați exemple.

❸ Identificați în *figura 1.2* toate componentele unei ferestre **Word**.

❹ **EXPLOREAZĂ!** Care este destinația unei panglici de meniuri? Ce instrumente conține această panglică?

❺ Numiți comenzile destinate citirii şi salvării documentelor. Când se utilizează aceste comenzi?

❻ Numiți comenzile care se utilizează pentru efectuarea operațiilor de editare a textelor: ştergerea, copierea, mutarea, înlocuirea etc.

❼ Cum se asigură accesul rapid la comenzile din meniuri? Dați exemple.

❽ **CERCETEAZĂ!** Aflați denumirile tuturor butoanelor din panglica de instrumente.

❾ **STUDIU DE CAZ!** Creați cu ajutorul aplicației **Notepad** fişierele ce conțin textele **Cântec**, **Amintiri din copilărie** şi **Peripețiile Alicei în Țara Minunilor**. Deschideți aceste fişiere în aplicația **Word**. Comparați modurile în care sunt afişate textele în aplicațiile **Notepad** şi **Word**.

CÂNTEC

Frumoasă eşti, pădurea mea,
Când umbra-i încă rară
Şi printre crengi adie-abia
Un vânt de primăvară...

Când de sub frunze moarte ies
În umbră viorele,
Iar eu străbat huceagul des
Cu gândurile mele...

Când strălucesc sub rouă grea
Cărări de soare pline,
Frumoasă eşti, pădurea mea,
Şi singură ca mine...

George Topîrceanu

AMINTIRI DIN COPILĂRIE

Stau câteodată și-mi aduc aminte ce vremi și ce oameni erau în părțile noastre pe când începusem și eu, drăgăliță-Doamne, a mă ridica băiețaș la casa părinților mei, în satul Humulești, din târg drept peste apa Neamțului; sat mare și vesel, împărțit în trei părți, care se țin tot de una: Vatra satului, Delenii și Bejenii.

Ș-apoi Humuleștii, și pe vremea aceea, nu erau numai așa, un sat de oameni fără căpătâiu, ci sat vechi răzeșesc, întemeiat în toată puterea cuvântului: cu gospodari tot unul ca unul, cu flăcăi voinici și fete mândre, care știau a învârti și hora, dar și suveica, de vuia satul de vatale în toate părțile; cu biserică frumoasă și niște preoți și dascăli și poporeni ca aceia, de făceau mare cinste satului lor...

Ion Creangă

PERIPEȚIILE ALICEI ÎN ȚARA MINUNILOR

Alice este o fetiță ca toate fetițele. Îi plac poveștile. Într-o zi adoarme cu gândul la ele și începe să viseze că se află într-o "țară a minunilor". Aici se petrec tot felul de întâmplări, cum numai în povești se pot întâmpla.

Astfel, de pildă, ea descoperă aici că se poate face mai mare sau mai mică, după nevoie.

Iat-o încercând să pătrundă într-o grădină fermecată, de care o despărțea însă o ușiță atât de scundă, încât nici capul nu-i putea intra. Ce-i de făcut? "Ah ce bine ar fi să mă pot strânge ca o lunetă", se gândea ea. Și tocmai atunci, găsește o sticluță pe eticheta căreia stătea scris cu litere frumoase și mari, de tipar: "BEA-MĂ".

Ușor de zis: "Bea-mă", dar Alice, o fetiță înțeleaptă, cuminte, nu era să se repeadă să facă una ca asta.

– Ba nu, zise ea, întâi să mă uit, să văd dacă nu cumva scrie pe sticlă: "Otravă".

Lewis Carroll

⓾ **EXERSEAZĂ!** Editaţi textele propuse de profesor folosind comenzile din meniul **Home** și butoanele respective din panglica de instrumente standard.

1.2. Formatarea caracterelor

Termeni-cheie:
- tehnoredactare
- font
- stil de afişare
- formatare a caracterelor

Scrisul a fost descoperit cu circa 5 000 de ani în urmă şi până în prezent este cel mai important mijloc de păstrare şi transmitere a informaţiei. Dezvoltarea civilizaţiei

umane este indisolubil legată de dezvoltarea tehnicii scrisului: scrierea de mână, inventarea tiparului, procesarea textelor cu ajutorul calculatorului.

Pregătirea tehnică şi grafică a unui manuscris înainte de a începe operația de tipărire se numeşte *tehnoredactare*.

Tehnoredactarea se realizează pe trei niveluri – pagină, paragraf, caracter – şi are drept scop crearea unor texte lizibile. La nivel de caracter, utilizatorul stabileşte sau modifică proprietățile fiecărui caracter, separat sau în componența unei porțiuni de text. În aplicația **Word** proprietățile principale ale unui caracter sunt:

– fontul;
– stilul de afişare;
– dimensiunea;
– culoarea de afişare;
– efectele speciale.

***Fontul* reprezintă un set complet de caractere (toate literele alfabetului, inclusiv majuscule şi minuscule, cifre, semne de punctuație şi simboluri) cu un aspect grafic unitar.**

În sistemul de operare **Windows**, fonturile se păstrează în fişiere cu extensia **.ttf** şi sunt utilizate de toate aplicațiile în care se prelucrează texte. Fişierele respective se elaborează de pictori, programatori şi specialişti în domeniul tehnicii tiparului. Fiecare font are o denumire şi este protejat prin dreptul de autor. În prezent sunt cunoscute circa 1 500 de fonturi. Descrierea fonturilor frecvent utilizate este prezentată în *tabelul 1.1*.

Tabelul 1.1

Fonturi frecvent utilizate

Denumirea fontului	Aspectul grafic
Arial	Conţine litere bine proporţionate, robuste şi clar conturate. Se foloseşte pentru titluri.
Courier Courier New	Imită aspectul caracterelor de la maşina de scris. Toate simbolurile au aceeaşi lăţime. Aplicaţii: tabele, corespondenţa comercială, rapoarte.
Times New Roman	Caracterele acestui font au la extremități o mică liniuță finală (serif) ce ghidează privirea de la o literă la alta. Seriful îmbunătățește lizibilitatea cuvintelor și le oferă o formă individuală. Se utilizează pentru texte mari: manuale, opere literare, publicații.
Wingdings (Sunete de aripi)	Include diverse simboluri și reprezentări grafice, utilizate ca ornamente sau semne de reprezentare:

Stilul de afişare specifică modul în care caracterele vor fi afişate pe ecran sau tipărite pe hârtie. Aplicaţia **Word** foloseşte următoarele stiluri de afişare:

Regular (Normal) – caracterele sunt afişate exact aşa cum au fost concepute de creatorii fontului;

Italic (Înclinat) – caracterele sunt înclinate la 15°;

Bold (Îngroşat) – grosimea liniilor care formează caracterele este mai mare;

Bold Italic (Îngroşat înclinat) – se aplică concomitent stilurile **Bold** şi **Italic**.

Aspectul grafic al stilurilor de afişare este prezentat în *tabelul 1.2.*

Tabelul 1.2

Stiluri de afişare

Stilul	Aspectul grafic			
Regular	Arial	Bookman	Courier	Times New Roman
Italic	*Arial*	*Bookman*	*Courier*	*Times New Roman*
Bold	**Arial**	**Bookman**	**Courier**	**Times New Roman**
Bold Italic	***Arial***	***Bookman***	***Courier***	***Times New Roman***

Dimensiunile caracterelor, conform unor tradiţii tipografice, se măsoară în **puncte**, un punct fiind egal cu 0,351 mm. Pentru exemplificare, în *tabelul 1.3* sunt prezentate dimensiunile frecvent utilizate în textele din manuale, opere literare, ziare, corespondenţa de afaceri.

Tabelul 1.3

Dimensiuni de caractere

Dimensiunea (*punctele*)	Aspectul grafic			
8	Arial	Bookman	Courier	Times New Roman
10	Arial	Bookman	Courier	Times New Roman
12	Arial	Bookman	Courier	Times New Roman
14	Arial	Bookman	**Courier**	Times New Roman
16	Arial	Bookman	**Courier**	Times New Roman

Caracterele unui text pot fi afişate pe ecran sau tipărite pe hârtie în diferite culori. Fiecărui caracter i se pot atribui şi efecte speciale. Unele efecte vor fi vizibile numai pe ecran (de exemplu, animaţii de genul *Luminile Las-Vegasului*), altele vor fi tipărite şi pe hârtie (de exemplu, tăierea cu o linie, scrierea ca indice superior sau inferior).

Numim *formatare a caracterelor* procesul de stabilire a proprietăţilor caracterelor unui text: fontul, stilul de afişare, dimensiunile, culoarea, efectele speciale.

Formatarea caracterelor (litere, numere, simboluri) se realizează prin intermediul meniului **Home**, opţiunea **Font**, la activarea căreia va apărea fereastra de dialog din *figura 1.3*.

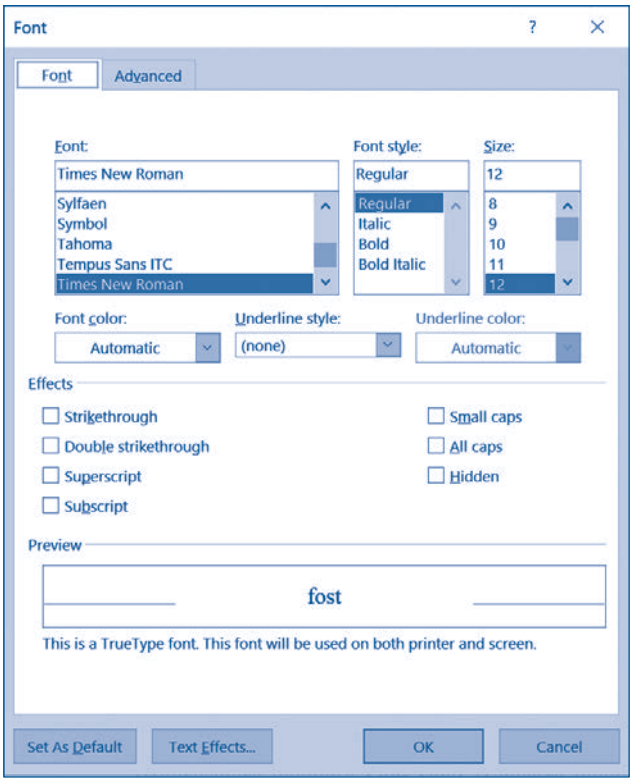


Fig. 1.3. Fereastra de dialog **Font**

Setările făcute în această fereastră vor acționa asupra porțiunii selectate de text sau, în lipsa unei astfel de selectări, asupra cuvântului în interiorul căruia se află punctul de inserare. Dacă punctul de inserare nu se află în interiorul niciunui cuvânt, setările din fereastra de dialog **Font** vor acționa asupra textului care se introduce de la tastatură.

Accesul rapid la unele opțiuni din fereastra de dialog **Font** se realizează prin intermediul panglicii de instrumente de formatare. Această panglică conține listele derulante **Font**, **Font Size** (Dimensiunea fontului), **Font color** (Culoarea fontului) şi butoanele **Bold**, **Italic**, **Undeline** (Subliniere), **Highlight** (Evidențiere). Menționăm că elementele de control din panglica de instrumente de formatare servesc şi ca indicatoare de proprietăți. Ele își schimbă starea în funcție de proprietățile caracterelor şi ale paragrafelor în care se află punctul de inserare.

Întrebări şi exerciții

❶ Explicați termenul *tehnoredactare*. Când şi cum se efectuează tehnoredactarea?

❷ **EXPLOREAZĂ!** Numiți proprietățile principale ale caracterelor. Cum pot fi aflate aceste proprietăți în cadrul aplicației **Word**?

❸ Explicați termenul *font*. Cum se păstrează fonturile în sistemul de operare **Windows**?

❹ Găsiți pe discul rigid dosarul **Font**. Afișați pe ecran şi tipăriți la imprimantă fonturile frecvent utilizate.

❺ Numiți stilurile de afișare utilizate în aplicația **Word**.

❻ **CERCETEAZĂ!** Determinați stilul de afișare a caracterelor din următoarele cuvinte:

<u>școală</u>	**clasă**	*manual*
automobil	calculator	**arbore**

Introduceți în calculator și formatați aceste cuvinte conform modelului de mai sus. Observați cum își schimbă starea butoanele **Bold**, **Italic** și **Underline** la trecerea punctului de inserare dintr-un cuvânt în altul.

❼ Explicați termenul *formatare*. Cum se realizează formatarea caracterelor în aplicația **Word**?

❽ **EXPLOREAZĂ!** Indicați în panglica de instrumente de formatare toate elementele de control destinate formatării caracterelor. Explicați cum se aplică aceste instrumente.

❾ **EXPERIMENTEAZĂ!** Introduceți în calculator și formatați textul conform modelului ce urmează:

Arial	**Arial**	*Arial*	***Arial***
Courier	**Courier**	*Courier*	***Courier***
Bookman	**Bookman**	*Bookman*	***Bookman***
Times	**Times**	*Times*	***Times***

Folosiți caractere cu dimensiunile 12, 14, 16 și 18 puncte. Observați cum își schimbă starea elementele de control din panglica de instrumente la trecerea punctului de inserare dintr-un cuvânt în altul.

❿ **EXERSEAZĂ!** Introduceți în calculator textele **Și dacă...**, **Glossă** și formatați-le conform modelului propus.

ȘI DACĂ...

Și dacă ramuri bat în geam
Și se cutremur plopii,
E ca în minte să te am
Și-ncet să te apropii.

Și dacă stele bat în lac
Adâncu-i luminându-l,
E ca durerea mea s-o-mpac
Înseninându-mi gândul.

Și dacă norii deși se duc
De iese-n luciu luna,
E ca aminte să-mi aduc
De tine-ntotdeauna.

Mihai Eminescu

GLOSSĂ

Vreme trece, vreme vine,
Toate-s vechi și nouă toate;
Ce e rău și ce e bine
Tu te-ntreabă și socoate;

Nu spera şi nu ai teamă,
Ce e val ca valul trece;
De te-ndeamnă, de te cheamă,
Tu rămâi la toate rece.

Multe trec pe dinainte,
În auz ne sună multe,
Cine ține toate minte
Și ar sta să le asculte?...

Tu așază-te de-o parte,
Regăsindu-te pe tine,
Când cu zgomote desarte
Vreme trece, vreme vine...

Mihai Eminescu

⓫ **EXERSEAZĂ!** Încărcați în aplicația **Word** textele **Amintiri din copilărie** și **Peripețiile Alicei...**. Formatați caracterele din aceste texte după cum urmează:
– **titlul:** Arial, 14 puncte, Bold;
– **textul de bază:** Times New Roman, 12 puncte, Regular;
– **numele autorului:** Courier New, 10 puncte, Italic.
Salvați documentele create pe disc.

⓬ **STUDIU DE CAZ!** Schimbați fontul **Times New Roman** din documentele **Amintiri din copilărie** și **Peripețiile Alicei...** în fontul **Courier New**. Care texte sunt mai lizibile? Cum se schimbă spațiul ocupat de fiecare text pe pagină?

⓭ **CREEAZĂ!** Găsiți cea mai reușită, după părerea dvs., variantă de formatare a caracterelor din textele **Cântec**, **Și dacă...**, **Glossă**. Numiți avantajele și dezavantajele variantelor examinate.

1.3. Formatarea paragrafelor

Termeni-cheie:
- paragraf
- aliniere
- indentare
- spațiere

Pentru a fi mai uşor înțelese, textele mari se împart în fragmente relativ mici, numite **paragrafe**. De obicei, fiecare paragraf cuprinde un anumit subiect, iar trecerea de la un paragraf la altul marchează schimbarea subiectului. În aplicația **Word**

sfârşitul unui paragraf se indică prin acționarea tastei <*Enter*>. Pe ecran, sfârşitul de paragraf este simbolizat prin semnul "¶". Proprietățile principale ale unui paragraf sunt:

– alinierea;
– indentarea;
– indentarea primei linii;
– spațiul dintre liniile paragrafului;
– spațiul de dinainte şi de după paragraf;
– efectele speciale la nivel de paragraf.

Alinierea caracterizează modul în care sunt aranjate pe orizontală liniile unui paragraf: la stânga, la dreapta, pe centru sau la ambele margini. Opțiunile de aliniere a paragrafelor sunt prezentate în *tabelul 1.4*.

Tabelul 1.4

Alinierea paragrafelor

Left (La stânga)	Fiecare linie a paragrafului începe din acelaşi punct în partea stângă şi se termină în diferite puncte în partea dreaptă, în funcție de numărul de caractere din linia respectivă. Paragrafele aliniate la stânga sunt considerate prietenoase şi deschise cititorului. Întrucât oamenii citesc de la stânga la dreapta, aceasta este cea mai potrivită alegere pentru textele mari.
Center (Pe centru)	Fiecare linie va fi aşezată în centru, între marginile din stânga şi dreapta ale paragrafului. Se foloseşte pentru titluri, formule şi fragmente scurte de texte. Nu trebuie folosit pentru pasaje lungi.
Right (La dreapta)	Liniile încep în diferite puncte în partea stângă şi se termină în acelaşi punct în partea dreaptă. Această aliniere se foloseşte pentru fragmentele scurte de text din partea stângă ale unor figuri.
Justified (La ambele margini)	Fiecare linie este încadrată exact între marginile din stânga şi dreapta ale paragrafului. În caz de necesitate, programul introduce în mod automat câteva spații suplimentare între cuvintele din fiecare linie. Paragraful aliniat la ambele margini este utilizat pentru a crea o imagine sobră sau clasică.

Indentarea (din engleză *indent* "adâncitură", "zimț") caracterizează poziția pe orizontală a marginilor fiecărui paragraf. Indentarea primei linii se referă la începutul acesteia în raport cu marginea din stânga a paragrafului. Opțiunile de indentare sunt prezentate în *tabelul 1.5*.

Tabelul 1.5

Indentarea paragrafelor

Left (Indentare stânga)	×× ×× Marginea din stânga a paragrafului este introdusă în interiorul textului. ×××××××××××××××××××××××× ×× ×× ×× ××
Right (Indentare dreapta)	×× ×× Marginea din dreapta a paragrafului este trasă în interiorul textului. ×××××××××××××××××××××××××× ××××× ×× ×× ×× ×× ×× ×× ×× ×× ××××××× ×× ××
Special (Indentare specială)	Începutul primei linii (**First line**) este deplasat în interiorul sau exteriorul paragrafului. Deplasarea în interiorul textului se foloseşte pentru marcarea începutului unui paragraf nou şi este cunoscută în limba română sub denumirea de **alineat**. Este recomandată pentru textele aliniate la ambele margini. ×××××××××××××××××××××××××××××× ×× Deplasarea în exterior (*hanging* "care atârnă afară") a începutului primei linii a fiecărui paragraf creează un contrast puternic cu marginea paragrafelor din componența listelor cu buline sau a listelor numerotate. ×××××××××××××××××××××× ×××

Spațiul dintre liniile fiecărui paragraf (**Line space**) este stabilit în mod automat în funcție de dimensiunile caracterelor utilizate. Totuşi unele documente pot fi prelucrate mai uşor când există mai mult spațiu. De exemplu, spațiul suplimentar este util atunci când cineva doreşte să facă unele corectări în paragraful respectiv.

Spațiul dinainte (**Before**) şi după (**After**) paragraf este un spațiu suplimentar care se adaugă înaintea primei şi, respectiv, după ultima linie a paragrafului. Aceste spații permit cititorului să găsească mai uşor începutul fiecărui paragraf, făcând textul mai lizibil. Spațiile suplimentare sunt recomandate pentru textele aliniate la stânga.

Numim *formatare a paragrafelor* procesul de stabilire a proprietăților paragrafelor unui text: alinierea, indentarea, spațierea, efectele speciale.

Formarea paragrafelor se realizează prin intermediul meniului **Format**, opțiunile **Paragraph** şi **Borders and Shading** (Chenare şi umbrire). La activarea opțiunii **Paragraph** se afişează fereastra de dialog din *figura 1.4*.

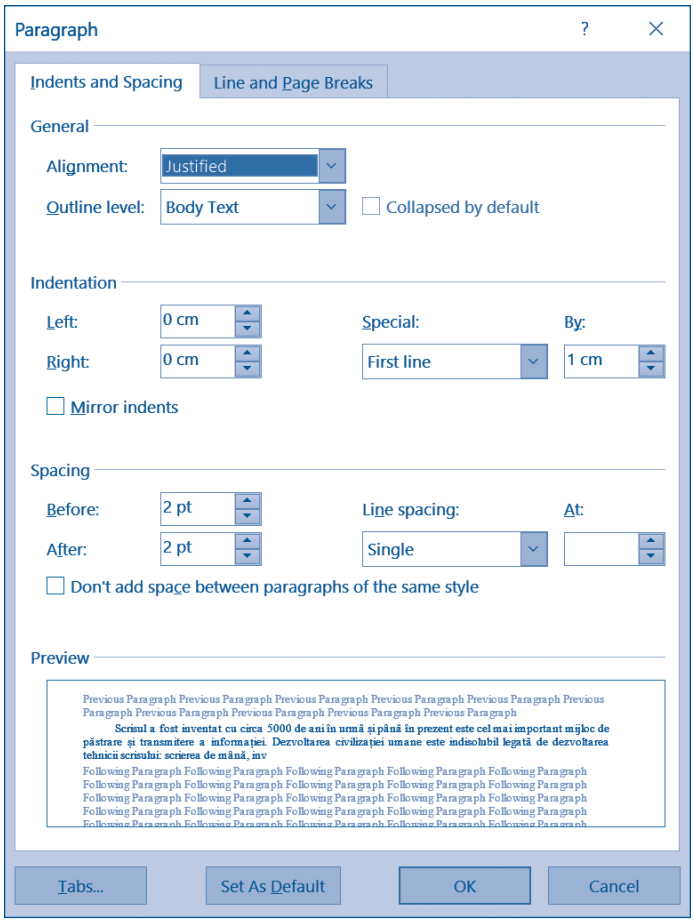


Fig. 1.4. Fereastra de dialog **Paragraf**

Setările făcute în această fereastră vor acționa asupra paragrafelor selectate sau, în lipsa unei astfel de selecții, asupra paragrafului în interiorul căruia se află punctul de inserare. Dacă punctul de inserare se află la începutul unui paragraf nou, setările din fereastra de dialog vor acționa asupra textului care se introduce de la tastatură.

Accesul rapid la opțiunile din fereastra de dialog **Paragraph** se realizează prin intermediul panglicii de instrumente de formatare şi al riglei orizontale. Panglica de instrumente conține butoanele **Align Left**, **Center**, **Align Right** şi **Justify**, destinate alinierii textului. Rigla orizontală conține cursoarele **First Line Indent** (▽), **Left Indent** (⌂) şi **Right Indent** (⌂), destinate indentării paragrafelor. Amintim că elementele de control din panglica de instrumente şi rigla orizontală servesc şi ca indicatoare de proprietăți.

La activarea opțiunii **Borders and Shading** se afişează fereastra de dialog din *figura 1.5*.

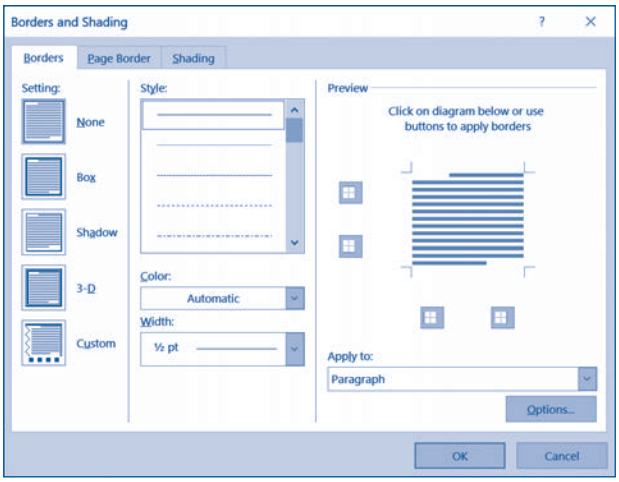


Fig. 1.5. Fereastra de dialog **Borders and Shading**

Această fereastră permite stabilirea unor efecte la nivel de paragraf: încadrarea paragrafelor în diverse tipuri de chenare, umbrirea paragrafelor (suprapunerea textului pe un fundal color) etc.

Întrebări şi exerciţii

❶ Care este scopul divizării textelor în paragrafe? Daţi exemple.

❷ Cum se delimitează paragrafele în aplicaţiile **Notepad** şi **Word**?

❸ **EXPLOREAZĂ!** Numiţi proprietăţile principale ale unui paragraf. Cum pot fi afişate aceste proprietăţi în cadrul aplicaţiei **Word**?

❹ Care sunt opţiunile de prezentare a liniilor unui paragraf? În ce cazuri se utilizează fiecare dintre aceste opţiuni?

❺ Care sunt opţiunile de indentare a marginilor unui paragraf? În ce cazuri se utilizează aceste opţiuni?

❻ **EXPERIMENTEAZĂ!** Cum se setează spaţiul dintre liniile unui paragraf şi dintre paragrafe?

❼ Explicaţi termenul *formatarea paragrafelor*.

❽ **CERCETEAZĂ!** Utilizând sistemul de asistenţă, aflaţi destinaţia tuturor elementelor de control din fereastra de dialog **Paragraph**. Observaţi cum îşi schimbă aspectul textul din zona de previzualizare a acestei ferestre.

❾ Indicaţi pe rigla orizontală şi în panglica de instrumente de formatare elementele de control destinate formatării paragrafelor. Explicaţi cum se aplică aceste instrumente.

❿ **CERCETEAZĂ!** Utilizând sistemul de asistenţă **Help**, aflaţi destinaţia tuturor elementelor de control din fereastra de dialog **Borders and Shading**. Observaţi cum îşi schimbă aspectul textul din zona de previzualizare a acestei ferestre.

⓫ Încărcaţi în aplicaţia **Word** textele **Amintiri din copilărie** şi **Peripeţiile Alicei...**. Formataţi paragrafele acestor texte conform modelelor prezentate pe pagina 8 a acestui manual. Verificaţi formatările cu ajutorul opţiunii "**?**" a sistemului de asistenţă.

⓬ Formataţi documentele **Cântec**, **Şi dacă...** conform modelelor prezentate pe paginile, respectiv, 7 şi 12 ale acestui manual. Afişaţi pe ecran proprietăţile caracterelor şi paragrafelor din documentele formatate.

1.4. Formatarea paginilor

Termeni-cheie:
- pagină fizică
- pagină logică
- antet şi subsol
- secțiune

În procesul tehnoredactării, aplicația **Word** divizează documentul în pagini. Când o pagină este completă, aplicația trece în mod automat la o pagină nouă. Pentru a trece forțat la o pagină nouă, fără a completa pagina curentă, se utilizează delimitatorul de pagină (**Page break**).

Proprietățile principale ale unei pagini sunt:
– dimensiunea fizică;
– orientarea;
– marginile;
– dimensiunile antetului şi ale subsolului;
– alinierea pe verticală a textului.

Structura paginii este prezentată în *figura 1.6.*

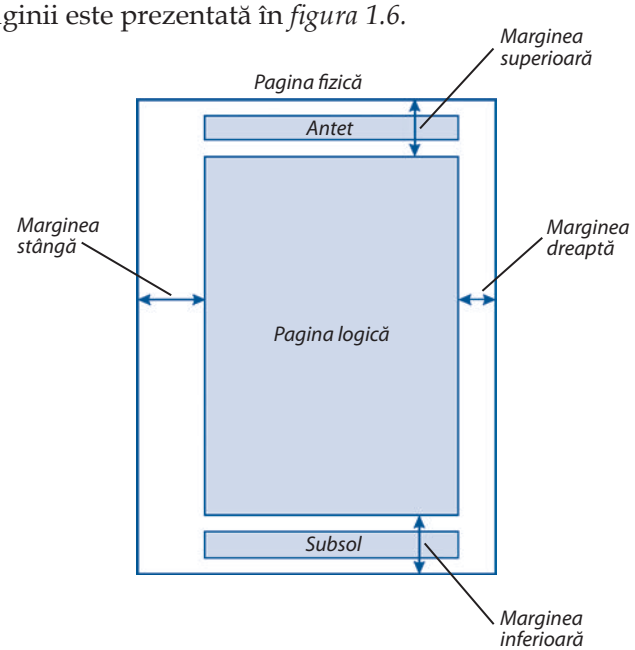


Fig. 1.6. Structura paginii

Numim *formatare a paginilor* procesul de stabilire a proprietăților paginilor din document: dimensiunea, orientarea, marginile, alinierea pe verticală, dimensiunile antetului şi ale subsolului.

Formatarea paginilor se realizează cu ajutorul comenzii **Page Layout** (Afişează în pagină, Definire pagină), la activarea căreia pe ecran apare fereastra de dialog din *figura 1.7.*

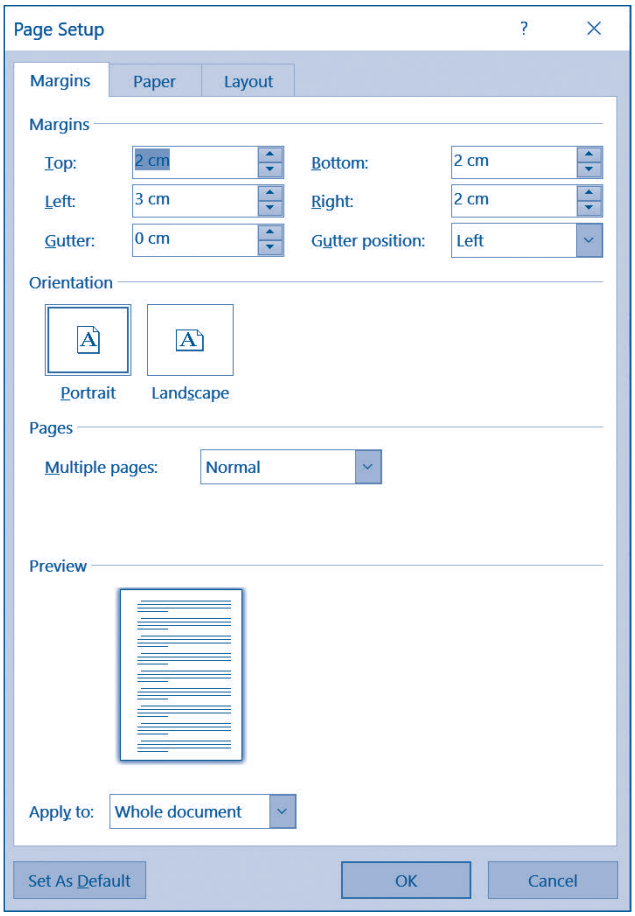


Fig. 1.7. Fereastra de dialog **Page Setup**

Elementele de control ale acestei ferestre au următoarea destinație:

Margins (Margini) – stabilirea dimensiunilor marginilor;

Paper Size (Dimensiunea hârtiei) – stabilirea dimensiunilor paginii fizice şi a orientării acesteia;

Paper Source (Sursa de hârtie) – indicarea modului de încărcare a hârtiei în imprimantă;

Layout (Aşezarea în pagină) – alinierea pe verticală a textului în cadrul paginii (în cazul în care pagina logică nu este complet acoperită de conținut).

Accesul rapid la unele opțiuni din pagina **Margins** se realizează prin intermediul riglelor orizontală şi verticală. Partea albă a fiecărei rigle arată dimensiunile paginii logice, iar partea întunecată indică distanța dintre marginea hârtiei şi marginea textului. Modificarea dimensiunii marginilor se efectuează "trăgând" linia ce desparte sectoarele albe şi întunecate ale riglei în direcția dorită.

Menționăm faptul că spațiile albe din jurul textului sau, cu alte cuvinte, marginile textului creează o imagine elegantă, prietenoasă şi deschisă a documentului. Totodată, marginile laterale oferă cititorului posibilitatea de a ține documentul în mână, fără să acopere textul sau imaginile.

Antetul şi **subsolul** sunt porţiuni de text ce apar în partea de sus sau de jos a fiecărei pagini din document (*fig. 1.6*). Antetele apar în partea de sus a paginii, deasupra textului de bază, iar subsolurile apar dedesubtul acestuia.

Crearea antetelor se efectuează cu ajutorul comenzii **Insert**, **Header** (Inserare, Antet), iar a subsolurilor – cu ajutorul comenzii **Insert**, **Footer** (Inserare, Subsol). La lansarea acestor comenzi, pe ecran apar liste ce conţin diverse modele din care utilizatorul poate alege pe cele mai potrivite pentru documentele lui (*fig. 1.8*)

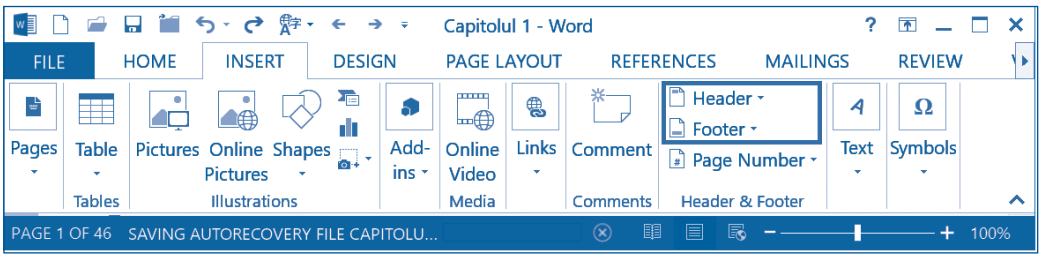


Fig. 1.8. Comenzile **Header** şi **Footer**

În afara modelelor de antete şi subsoluri, listele **Header** şi **Footer** conţin de asemenea butoane care permit inserarea automată în antet şi subsol a numelui de fişier, a datei şi a orei curente, a numărului total de pagini din document etc.

Accentuăm faptul că formatarea paginilor este o operaţie cu o întindere largă, care se poate extinde asupra tuturor paginilor din document. La procesarea unor documente complexe pot apărea însă situaţii când sunt necesare două sau mai multe formate de pagină. De exemplu, un manual conţine foaia de titlu, textul de bază, anexele, bibliografia, cuprinsul etc. Stabilirea diferitor formate de pagină se realizează cu ajutorul secţiunilor.

***Secţiunile* reprezintă zone continue din document în cadrul cărora se pot stabili diferite formate de pagină.**

Pentru a trece la o secţiune nouă, se utilizează delimitatorul de secţiune (**Section Break**). Inserarea delimitatorului de pagină şi a delimitatorului de secţiune se efectuează cu ajutorul comenzii **Page Layout**, **Breaks** (Aşezarea în pagină, Delimitatoare), la activarea căreia apare fereastra de dialog din *figura 1.9*.

Această fereastră permite inserarea delimitatorului de secţiune în diferite locuri ale documentului: la începutul unei pagini noi (**Next Page**), pe pagina curentă în locul unde se află cursorul (**Continuous**), la începutul unei pagini noi cu număr par (**Even Page**) sau impar (**Odd Page**). După inserarea delimitatorilor necesari, utilizatorul poate stabili formate de pagină pentru fiecare secţiune aparte. Paginile pot fi numerotate automat cu ajutorul comenzii **Insert**, **Page Number** (Inserare, Număr de pagină).

Întrebări şi exerciţii

1. Care este destinaţia delimitatorului de pagină? Cum credeţi, conţine acest manual delimitatori de pagină?
2. Numiţi proprietăţile principale ale paginilor unui document.
3. **CERCETEAZĂ!** Determinaţi următoarele proprietăţi ale paginilor din acest manual: dimensiunea fizică, orientarea, dimensiunea marginilor, alinierea pe verticală a textului.

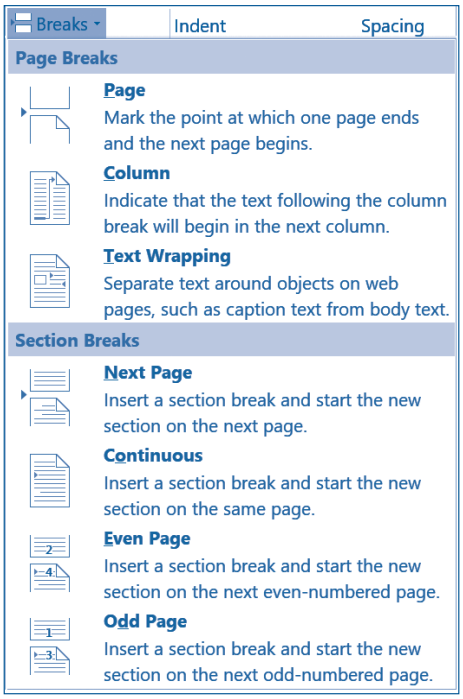


Fig. 1.9. Fereastra de dialog **Breaks**

❹ Care este destinația delimitatorului de secțiune? Cum credeți, conține acest manual delimitatori de secțiune? Argumentați răspunsul dvs.

❺ **EXPLOREAZĂ!** Găsiți pe riglele orizontală și verticală elementele de control destinate formatării paginilor.

❻ **EXPLOREAZĂ!** Utilizând sistemul de asistență, aflați destinația tuturor opțiunilor oferite de ferestrele de dialog **Page Setup**, **Breaks** și listele **Header**, **Footer**.

❼ Care este destinația antetului și subsolului? Ce informație pot conține aceste obiecte?

❽ **EXERSEAZĂ!** Formatați paginile documentelor create anterior conform modelelor propuse de profesor. Antetul fiecărei pagini va conține în partea stângă data și ora curentă, iar subsolul – denumirea fișierului, prenumele și numele elevului.

❾ **CREEAZĂ!** Comasați într-un document nou toate documentele elaborate anterior: **Cântec**, **Amintiri din copilărie**, **Peripețiile Alicei...**, **Și dacă...**, **Glossă**. Documentul nou-creat trebuie să păstreze toate formatările de pagină, de paragraf și de caractere ale documentelor inițiale. Numerotați paginile documentului creat.

❿ Explicați termenul *formatarea paginilor* și indicați elementele de control folosite în acest scop.

⓫ **CERCETEAZĂ!** Pot fi oare incluse într-un singur document pagini cu orientarea *portret* și pagini cu orientarea *peisaj*?

⓬ **EXPERIMENTEAZĂ!** Pentru a ușura cititul, textul documentului poate fi așezat în pagină în formă de coloane (**Columns**). Împărțirea în coloane se face prin selectarea porțiunii dorite de text și activarea comenzii **Page Layout**, **Columns**.
Copiați fișierele **Amintiri din copilărie**, **Peripețiile Alicei...** într-un singur document și împărțiți textul în coloane.

1.5. Liste și tabele

Termeni-cheie:
- listă
- element al listei
- tabel
- celulă

Pentru a evidenția paragrafele din text, care sunt legate printr-un subiect comun, se utilizează listele. De exemplu, la sfârşitul fiecărei teme din acest manual este inclusă o listă ce conține întrebări şi exerciții.

> ***Lista* reprezintă un fragment de text în care începutul fiecărui paragraf este evidențiat cu un semn special sau cu un număr de ordine. Paragrafele respective se numesc *elemente ale listei*.**

Pentru a crea o listă cu semne de evidențiere, se selectează paragrafele dorite şi se activează comanda **Home**, **Bullets** (Acasă, Semne de evidențiere). Comanda afişează pe ecran fereastra de dialog **Bullet Library** (Biblioteca de semne de evidențiere), prezentată în *figura 1.10*.

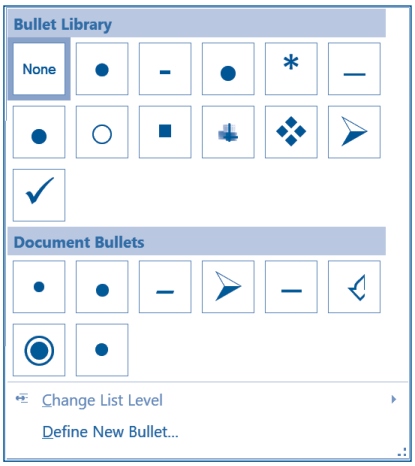


Fig. 1.10. Fereastra de dialog **Bullet Library**

Elementele de control ale acestei ferestre permit alegerea semnelor de evidențiere care vor marca fiecare element din listă.

Dacă se doreşte utilizarea altor semne de evidențiere, se va acționa butonul **Define New Bullet** (Definirea unui nou semn de evidențiere). În consecință, pe ecran va fi afişată o altă fereastră de dialog în care utilizatorul poate alege cele mai diverse semne de evidențiere, cum ar fi simboluri, pictograme, caractere ş.a.m.d.

Într-un mod similar se creează şi listele numerotate. La lansarea comenzii **Home**, **Numbering** (Acasă, Numerotare), pe ecran se afişează fereastra de dialog **Numbering Library** (Biblioteca de numerotări), care conține diverse variante de formatare a listelor în curs de creare sau de formatare (*fig. 1.11*).

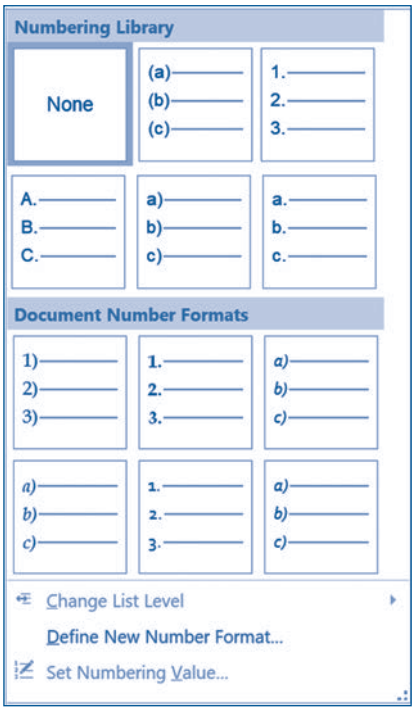


Fig. 1.11. Fereastra de dialog **Numbering Library**

Pentru a prezenta într-un mod lizibil un volum mare de date omogene, se utilizează tabelele.

***Tabelul* este un obiect complex format din rânduri şi coloane. Dreptunghiurile formate de intersecția unui rând cu o coloană se numesc *celule*.**

Fiecare celulă este relativ independentă de celelalte şi poate conține:
– texte;
– numere;
– imagini;
– formule;
– obiecte create cu alte aplicații.
Structura tabelelor din aplicația **Word** este prezentată în *figura 1.12*.

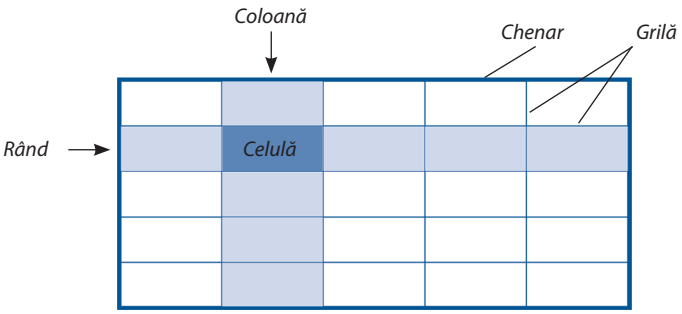


Fig. 1.12. Structura tabelelor

Cea mai simplă metodă de a crea un tabel este folosirea comenzii **Insert**, **Table** (Inserare, Tabel). La activarea acestei comenzi, pe ecran se afişează fereastra de dialog din *figura 1.13*.

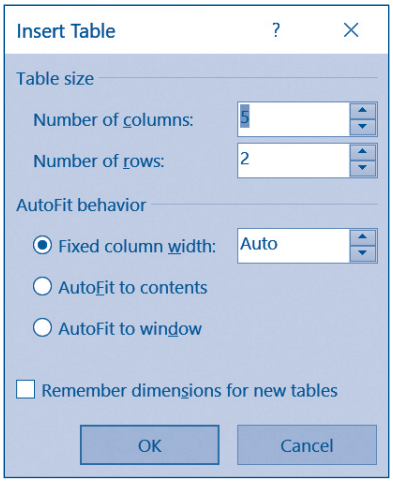


Fig. 1.13. Fereastra de dialog **Insert Table**

Contoarele din această fereastră permit stabilirea numărului de rânduri (**Number of rows**) şi a numărului de coloane (**Number of columns**) ale tabelului. După inserarea tabelului, utilizatorul poate introduce în celulele acestuia informaţiile dorite.

Asupra unui tabel pot fi efectuate următoarele **operaţii**:

– ştergerea;
– copierea;
– împărţirea unui tabel în două părţi;
– redimensionarea celulelor, rândurilor sau a coloanelor;
– adăugarea unui rând sau a unei coloane;
– unirea a două celule;
– împărţirea unei celule în două sau în mai multe celule.

Majoritatea comenzilor ce realizează aceste operaţii sunt incluse în meniurile **Design** şi **Layout**, care apar în panglica de meniuri imediat cum este selectat tabelul supus prelucrării (*fig. 1.14*).

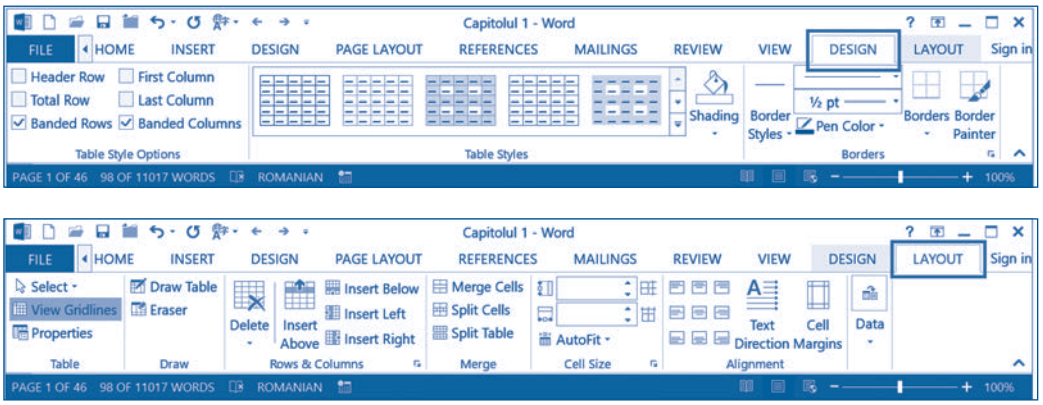


Fig. 1.14. Meniurile **Design** şi **Layout**

Accesul rapid la unele comenzi de prelucrare a tabelelor se realizează prin intermediul butoanelor din panglica de instrumente de formatare sau din meniurile contextuale. Meniurile respective conțin comenzi (instrumente) de desenare a grilei şi de aliniere a conținutului din fiecare celulă: creion, radieră, umplerea cu culoare, modele de linii, alinierea pe verticală etc. Menționăm că instrumentele respective pot fi utilizate şi pentru modificarea tabelelor create anterior.

Aspectul tabelelor depinde foarte mult de modul în care sunt separate celulele şi informațiile plasate în ele. În procesul creării şi formatării tabelelor, se vor respecta următoarele reguli:

1. Rândurile şi coloanele tabelului pot fi separate de liniile grilei sau de culorile pentru fundal.

2. Pentru anteturile sau titlurile de rânduri şi coloane, respectiv pentru informațiile conținute în celule, se vor utiliza diferite stiluri de caractere şi aliniere.

3. Se va evita înghesuirea textului între liniile alăturate ale grilei sau extinderea textului prea aproape de textul din coloanele alăturate.

Să reținem că utilizarea unor tabele, grila şi chenarul cărora sunt invizibile, permite alinierea numerelor şi cuvintelor în coloane precise, aranjarea paragrafelor unul lângă altul, amestecarea textului şi a imaginilor.

Întrebări şi exerciții

❶ Explicați termenii *listă* şi *element al listei*. Pentru ce se utilizează listele?

❷ **CERCETEAZĂ!** Găsiți în manual fragmentele de text care formează liste. Cum este evidențiat fiecare element al listei?

❸ **ELABOREAZĂ!** Creați un document ce conține lista elevilor din clasa dvs.

❹ **EXERSEAZĂ!** Afişați pe ecran şi tipăriți la imprimantă lista ce urmează.

Termeni-cheie:
- document;
- structura documentului;
- ferestre de document.

❺ Care este destinația tabelelor? Numiți elementele constitutive ale unui tabel.

❻ **CERCETEAZĂ!** Determinați numărul de rânduri, numărul de coloane şi numărul de celule ale *tabelelor 1.1*, *1.2* şi *1.4* din acest capitol. Observați cum sunt formatate antetele de coloană şi informațiile conținute în celule.

❼ Afişați pe ecran şi tipăriți la imprimantă *tabelele 1.2* şi *1.3* din acest capitol.

❽ **STUDIU DE CAZ!** Cum credeți, care sunt avantajele şi dezavantajele celor două metode de creare a tabelelor: inserare şi desenare?

❾ **EXPLOREAZĂ!** Găsiți în panglica de meniuri comenzile destinate creării şi formatării tabelelor. Explicați cum se utilizează aceste instrumente.

❿ **CERCETEAZĂ!** Utilizând sistemul de asistență, aflați destinația fiecărei opțiuni din meniurile **Design** şi **Layout**.

⓫ Afişați pe ecran stilurile de formatare a tabelelor oferite de opțiunea **Table**, **Styles**.

⓬ **CREEAZĂ!** Folosind un tabel, chenarul şi grila căruia sunt invizibile, creați documentul ce urmează (p. 26).

LISTA
participanţilor la Olimpiada de Informatică

1.	Munteanu Ion	Raionul Orhei, satul Chiperceni	tel.: 23-619
2.	Petrescu Angela	Bucureşti, Calea Dorobanţilor, nr. 3, ap.15	tel.: 261-32-64
3.	Chiriac Petru	Raionul Sângerei, satul Copăceni	tel.: 16-215
4.	Matei Ludmila	Chişinău, str. Cuza-Vodă, nr. 3, ap. 21	tel.: 32-65-84

ÎNVAŢĂ SĂ ÎNVEŢI! În afară de listele studiate în acest paragraf, aplicaţia **Word** oferă utilizatorului posibilitatea să creeze liste, ale căror elemente se pot afla pe mai multe niveluri (**Multilevel List**). Utilizând sistemul de asistenţă, aflaţi cum pot fi create astfel de liste.
Elaboraţi o listă care conţine pe primul nivel mărcile de automobile, iar pe al doilea – câteva modele de automobile de marca respectivă. De exemplu, *Dacia* este o marcă de automobile, iar *Dokker*, *Duster*, *Logan*, *Lodgy*, *Sandero* sunt modelele acestei mărci. Alte mărci de automobile sunt: *Ford*, *Mazda*, *Toyota*, *Volkswagen* ş.a.m.d.

1.6. Inserarea obiectelor

Termeni-cheie:
- inserare prin memoria-tampon
- inserare prin apelul altor aplicaţii
- inserare cu ajutorul fişierului

Este cunoscut faptul că într-un document pot fi inserate diverse obiecte: tabele, imagini, secvenţe sonore, secvenţe video. În funcţie de modul de creare a obiectelor care pot fi incluse într-un document, deosebim:

1) obiecte create în cadrul aplicaţiei **Word**;

2) obiecte create cu ajutorul altor aplicaţii.

De exemplu, din prima grupă fac parte tabelele care sunt create cu ajutorul opţiunilor din meniul **Insert**, **Table**. Din grupa a doua fac parte desenele create cu ajutorul aplicaţiei **Paint**, secvenţele sonore înregistrate cu ajutorul aplicaţiei **Voice Recorder** etc.

Sistemul de operare **Windows** oferă următoarele facilităţi pentru schimbul de obiecte între aplicaţii:

– prin intermediul memoriei-tampon (**Clipboard**);

– prin apelul aplicaţiei în care se creează obiectul dorit;

– cu ajutorul fişierelor.

Inserarea obiectelor prin intermediul memoriei-tampon presupune derularea concomitentă a aplicaţiei **Word** şi a altor aplicaţii, de exemplu **Calculator**, **Paint**,

Voice Recorder. Pentru a insera un obiect, se parcurg următorii paşi:

– se trece în aplicația în care va fi creat obiectul, de exemplu în aplicația **Paint**;

– obiectul dorit sau o parte din el se copie în memoria-tampon (comanda **Home**, **Copy**);

– se revine în aplicația **Word**;

– obiectul din memoria-tampon se include în document (comanda **Home**, **Paste**).

Inserarea prin intermediul memoriei-tampon este utilă atunci când din obiectul-sursă se preia numai o parte. De exemplu, dintr-un desen creat în aplicația **Paint** pot fi preluate numai anumite zone ale acestuia: cerul, soarele, o casă la orizont, un automobil etc.

Inserarea prin apelul altor aplicații se foloseşte atunci când obiectul ce trebuie inserat încă nu există. Pentru a apela aplicația în care se va crea obiectul dorit, se activează comanda **Insert**, **Object** (Inserare, Obiect). Fereastra de dialog afişată la activarea acestei comenzi este prezentată în *figura 1.15*.

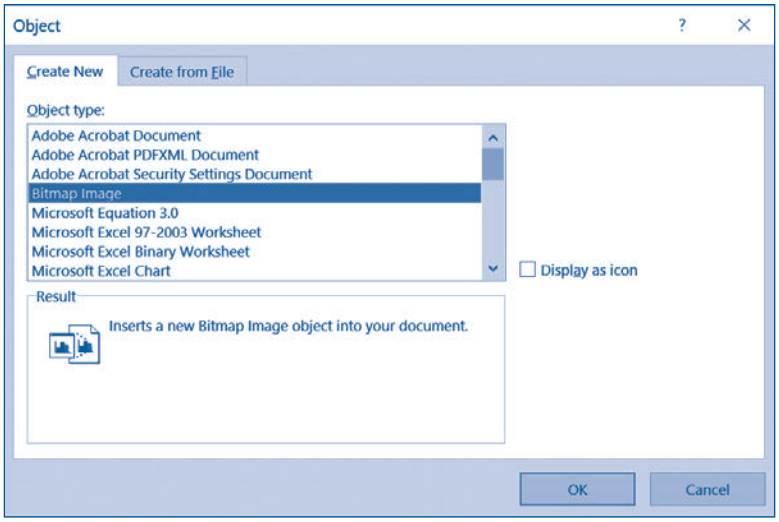


Fig. 1.15. Fereastra de dialog **Object**

Pagina **Create New** (Creează un obiect nou) a ferestrei în cauză conține lista **Object type** (Tipul obiectului) în care utilizatorul trebuie să indice tipul obiectului pe care doreşte să-l creeze. În funcție de tipul selectat, automat va fi lansată aplicația respectivă. De exemplu, la selectarea tipului **Bitmap Image** (Imagini în formatul hartă de biți) va fi lansată aplicația **Paint**. În continuare, utilizatorul poate lucra în această aplicație fără să părăsească aplicația **Word**. După înregistrarea şi, eventual, editarea imaginii, se revine în aplicația **Word** (*fig. 1.16*, p. 28).

După cum se vede din *figura 1.15*, lista **Object type** include mai multe tipuri de obiecte ce pot fi incluse într-un document:

Adobe Acrobat Document – documente în format portabil;

Bitmap Image – imagini în formatul *hartă de biți*;

Microsoft Equation – formule şi ecuații;

Microsoft ArtPowerPoint Slide – diapozitive;

Microsoft Graph Chart – grafice şi diagrame.

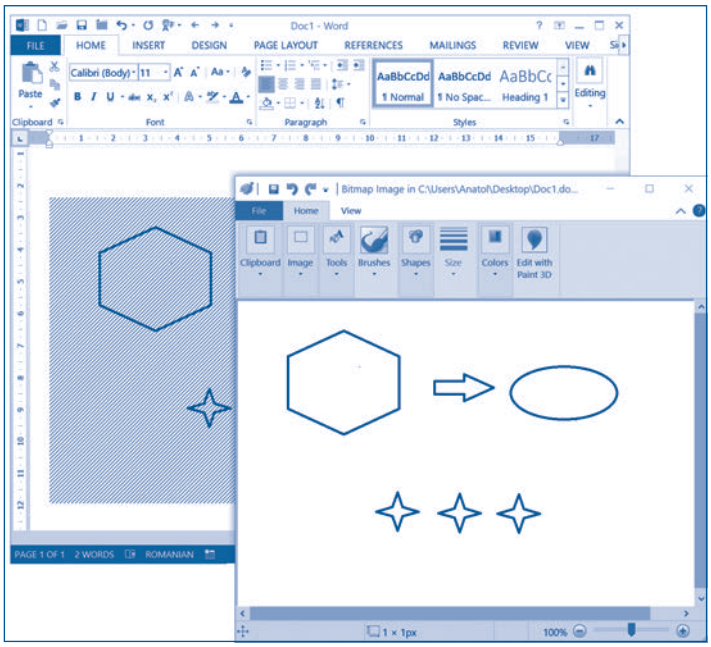


Fig. 1.16. Inserarea imaginilor create în aplicația **Paint**

Crearea obiectelor cu ajutorul acestor aplicații necesită o studiere prealabilă a meniurilor şi instrumentelor respective. De obicei, fiecare utilizator studiază numai aplicațiile frecvent folosite în activitatea profesională. De exemplu, un pictor va studia numai aplicațiile destinate prelucrării desenelor, iar un matematician – aplicațiile destinate inserării şi editării formulelor, graficelor şi diagramelor.

Inserarea cu ajutorul fişierelor se foloseşte atunci când obiectele respective sunt deja create, posibil, chiar şi de alți utilizatori. Pentru a insera un astfel de obiect, se foloseşte pagina **Create from File** (Inserare din fişier) a ferestrei **Object** (*fig. 1.17*).

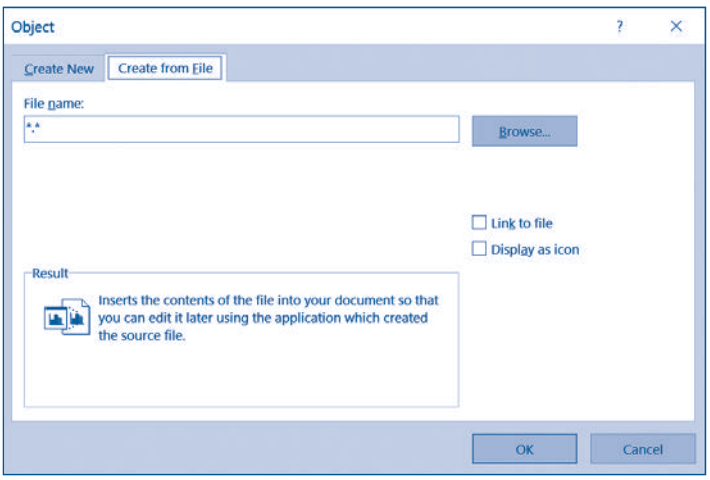


Fig. 1.17. Pagina **Create from File**

Această pagină conține următoarele elemente de control:

Browse (Răsfoiește) – permite selectarea fișierului ce conține obiectul dorit.

Link to file (Legătura cu fișierul) – stabilește o conexiune între fișier și document.

Când această casetă nu este marcată, fișierul selectat devine parte componentă a documentului **Word**. În afișarea documentului propriu-zis pe ecran, fișierul respectiv este simbolizat printr-o pictogramă. La efectuarea unui clic pe această pictogramă, procesorul de texte **Word** lansează în mod automat aplicația asociată cu extensia fișierului în cauză. Când caseta **Link to file** este marcată, în documentul **Word** nu se inserează fișierul propriu-zis, ci doar o legătură la el. La modificarea fișierului, automat va fi modificat și obiectul inclus în document. Opțiunea dată se utilizează atunci când la elaborarea unui document lucrează mai mulți utilizatori, fiecare dintre ei fiind responsabil de anumite obiecte. De exemplu, la elaborarea unui manual pictorul și autorul textului lucrează concomitent. Pictorul poate modifica desenul fără să intervină direct în documentul cu care lucrează autorul.

Display as icon (Afișează ca pictogramă) – obiectul inserat va fi reprezentat în document printr-o pictogramă. Se folosește pentru a economisi timpul și spațiul de afișare în cazurile în care se lucrează numai cu textul din document.

Pentru exemplificare, în *figura 1.18* este prezentat un document **Word** în care sunt inserate trei fișiere.

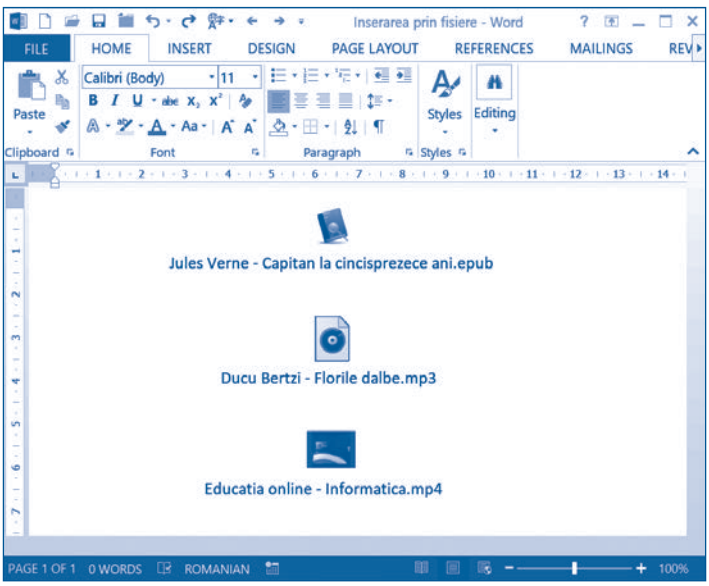


Fig. 1.18. Fișiere inserate în documentul **Word**

Primul dintre fișierele inserate în documentul **Word** din exemplul de mai sus, denumit "Jules Verne – Căpitan la cincisprezece ani", este o carte electronică în formatul **.epub**, utilizat pe larg pentru citirea cărților cu ajutorul dispozitivelor digitale portabile.

Al doilea fișier, denumit "Ducu Bertzi – Florile dalbe", reprezintă un cântec pe motive folclorice. Acest fișier audio este în formatul **.mp3**. Un fișier în format **.mp3** este de aproape zece ori mai mic decât fișierul audio original, dar sunetul are aproape aceeași calitate ca și pe discurile optice. Formatul **.mp3** constituie cel mai popular mijloc de stocare a fișierelor cu muzică atât pe calculatoare, cât și pe dispozitivele digitale portabile.

Fişierul "Educaţia online – Informatica" este în formatul **.mp4**, care se utilizează frecvent pentru a stoca informaţii video şi audio. Acest fişier conţine o lecţie televizată, destinată elevilor din clasele gimnaziale.

Menţionăm faptul că încorporarea în documentele **Word** a fişierelor multimedia poate duce la creşterea semnificativă a mărimii acestora. De exemplu, în cazul filmelor video, această creştere poate fi de ordinul gigaocteţilor. Evident, deschiderea şi prelucrarea unor astfel de fişiere poate crea mari dificultăţi, în special în cazul unor calculatoare cu performanţe mai modeste. Desigur, şi transmiterea unor astfel de documente prin reţelele de calculatoare va necesita mult timp.

Prin urmare, în cazul fişierelor multimedia, o soluţie mai rezonabilă ar fi stocarea acestora pe un disc al calculatorului personal sau pe un disc virtual, oferit de serviciile de stocare **Web** şi inserarea în documentele **Word** doar a legăturilor la ele.

Întrebări şi exerciţii

❶ Cum se clasifică obiectele ce pot fi inserate într-un document?

❷ Explicaţi metoda de inserare a obiectelor prin intermediul memoriei-tampon. În care cazuri se foloseşte această metodă?

❸ **CREEAZĂ!** Utilizând metoda de inserare a obiectelor prin intermediul memoriei-tampon, creaţi următorul document.

Semnificaţia pictogramelor

Calculatorul meu

Manualul de informatică

Cutia de reciclare

Pentru a aranja textul în pagină, folosiţi un tabel cu grila invizibilă.

❹ Explicaţi metoda de inserare a obiectelor prin apelul altor aplicaţii. Când se foloseşte această metodă?

❺ **CREEAZĂ!** Utilizând metoda de inserare prin apelul altor aplicaţii, creaţi documentul ce urmează:

Cifrele sistemului zecimal: 

Literele alfabetului român: 

Prima secvenţă sonoră va include denumirile cifrelor din sistemul zecimal: *zero*, *unu*, *doi*, …, *nouă*. Secvenţa a doua va include denumirile literelor alfabetului român: *a*, *be*, *ce*, …, *zet*.

❻ **EXPLOREAZĂ!** Afişaţi pe ecran ferestrele de dialog ale comenzilor **Shapes** (Figuri predesenate) şi **SmartArt** (Artă inteligentă) ale meniului **Insert**. Cum credeţi, în ce tip de documente pot fi folosite aceste obiecte?

❼ Explicaţi metoda de inserare a obiectelor cu ajutorul fişierelor. În care cazuri se foloseşte această metodă?

❽ Care este semnificația casetelor de marcare din ferestrele **Object** și **Create from File** (*fig. 1.15* și *1.17*)?

❾ **LUCREAZĂ ÎN ECHIPĂ!** Rugați colegul dvs. să creeze la un alt calculator imaginea din *figura 1.16*. Preluați fișierul respectiv prin rețea și inserați-l într-un document ce conține următorul text:

> Această pagină permite inserarea obiectelor cu ajutorul fișierelor. La acționarea butonului **Browse** (Răsfoiește), se afișează caseta de dialog cu același nume.

După acest text, inserați în document fereastra de dialog **Browse**.

❿ **ÎNVAȚĂ SĂ ÎNVEȚI!** Meniul **Insert** conține mai multe comenzi destinate inserării obiectelor decât cele explicate în acest paragraf. Utilizând sistemul de asistență, aflați destinația următoarelor comenzi: **Online Pictures** (Imagini online), **Screenshot** (Captură de ecran), **Online Video**, **Hyperlink**.

1.7. Formatarea imaginilor

Termeni-cheie:
- inserare a imaginilor
- proprietăți ale imaginilor
- operații asupra imaginilor

Documentele care conțin numai texte nu atrag atenția cititorului. Imaginile sunt mult mai sugestive şi, în majoritatea cazurilor, pot înlocui pagini întregi de text. În principiu, imaginile pot fi inserate ca şi oricare alte obiecte: secvențe sonore, formule, secvențe video etc. Întrucât în cazul documentelor tipărite imaginile au o importanță deosebită, aplicația **Word** conține instrumente speciale destinate inserării şi formatării imaginilor.

Pentru a indica sursele din care se preiau imaginile de inserat, se folosesc următoarele comenzi din meniul **Insert**:

Pictures (Poze) – din fişierul indicat de utilizator. Acest fişier se poate afla pe calculatorul local sau pe unul dintre calculatoarele conectate la rețea. Evident, pentru a accesa acest calculator, utilizatorul trebuie să aibă împuternicirile respective;

Online Pictures (Poze online) – din Internet sau de pe discurile virtuale ale utilizatorului;

Shapes – din biblioteca de obiecte grafice predesenate (din limba engleză *shape* "formă, figură");

SmartArt – din aplicația **Microsoft Smart Art** (Artă inteligentă);

Chart – din aplicația **Microsoft Graph Chart** (*chart* "diagramă, grafic").

Imaginile inserate în document se caracterizează prin următoarele proprietăți:
– culoare, luminozitate şi contrast;
– poziție;
– dimensiuni;
– modul de aranjare a textului în jurul său;
– chenare şi umbre.

Numim *formatare a imaginilor* procesul de stabilire a proprietăților acestora.

Pentru a formata o imagine, mai întâi se selectează obiectul dorit. În funcție de tipul obiectului selectat, va apărea meniul **Picture Tools Format** (Instrumente pentru formatarea imaginilor), care conține mai multe comenzi destinate formatării imaginilor (*fig. 1.19*). Accentuăm faptul că multe dintre aceste comenzi se regăsesc şi în meniul contextual, care poate fi afişat executând cu clic-dreapta pe imaginea de formatat.

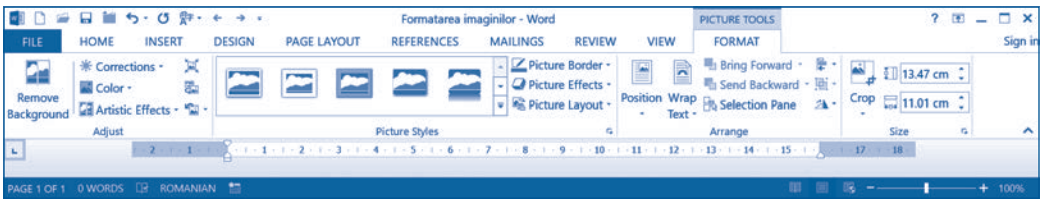


Fig. 1.19. Meniul **Picture Tools Format**

Comenzile din meniurile destinate formatării imaginilor permit stabilirea caracteristicilor de afişare a acestora: culoarea (**Color**), luminozitatea (**Brightness**), contrastul (**Contrast**), chenarul (**Picture Border**) etc. În caz de necesitate, marginile imaginii pot fi decupate (**Crop**).

Poziția imaginii în cadrul documentului se stabileşte cu ajutorul paginii **Position** din fereastra de dialog **Layout** (*fig. 1.20*).

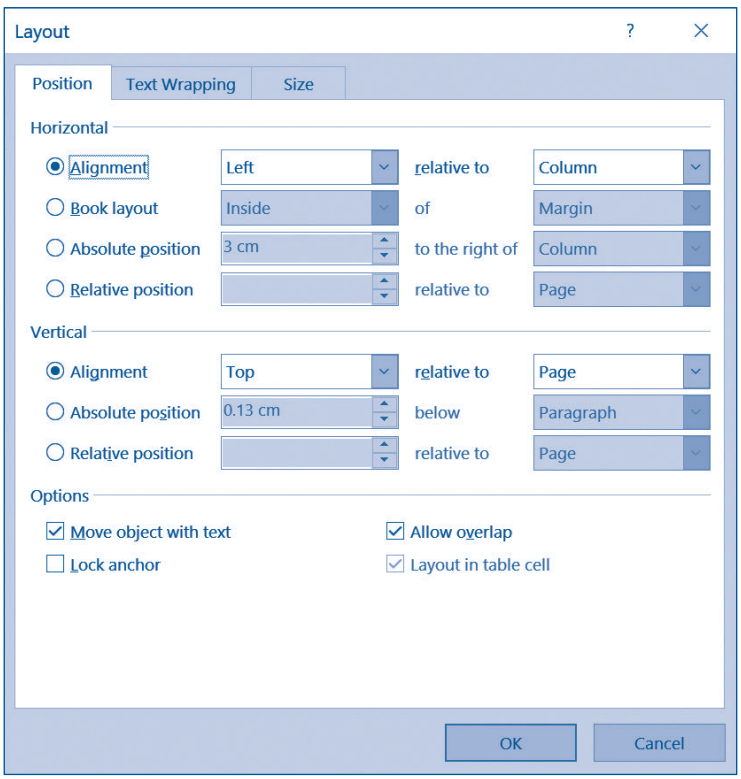


Fig. 1.20. Pagina **Position**

Elementele de control ale acestei pagini au următoarea destinație:

Alignment (Alinierea) – alinierea imaginii pe orizontală şi verticală, utilizând în calitate de repere, respectiv, marginile laterale şi verticale ale paginii sau ale coloanelor de text.

Absolute position (Poziția absolută) – poziția imaginii este definită în unitățile de măsură, utilizate de sistemul de operare sau de aplicația **Word**, de obicei în centimetri.

Relative position (Poziția relativă) – poziția imaginii este definită în unitățile relative, de obicei în procente, din dimensiunile paginii.

Move object with text (Mută obiectul împreună cu textul). Marcarea acestei casete leagă imaginea cu un anumit paragraf din text, legătura respectivă fiind simbolizată printr-o ancoră. Mutarea paragrafului în care este ancorată imaginea implică repoziționarea automată a acesteia. De exemplu, figurile din manualul dat sunt ancorate de paragrafele în care sunt menționate pentru prima oară.

Lock anchor (Blocarea ancorei). Când această casetă este dezactivată, utilizatorul poate reancora imaginea în alt paragraf. Marcarea casetei va imobiliza ancora în paragraful respectiv, evitându-se astfel erorile de poziționare.

Modul de aranjare a textului în jurul unei imagini se stabileşte cu ajutorul paginii **Text Wrapping** (Învelirea cu text), prezentată în *figura 1.21*.

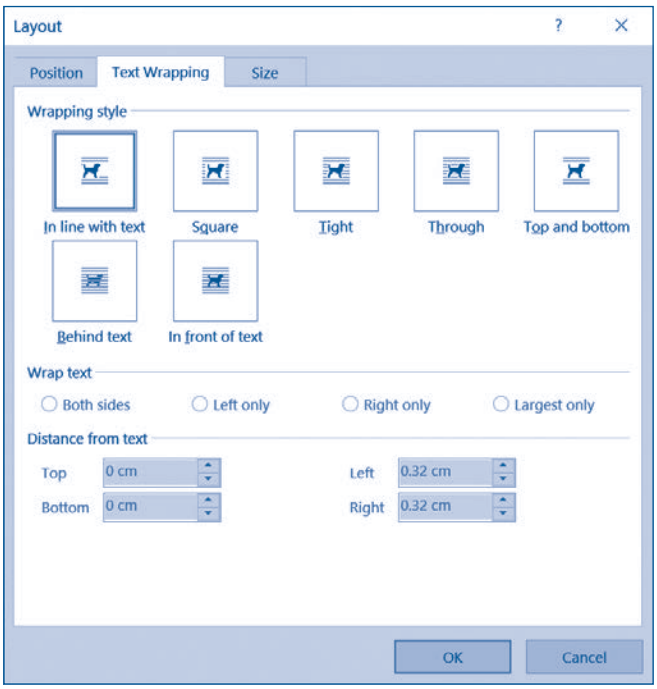


Fig. 1.21. Pagina **Text Wrapping**

Rubrica **Wrapping style** (Stilul de învelire) a acestei pagini permite selectarea stilului de aranjare a textului în jurul paginii.

In line with text – în linie cu textul. În acest caz imaginea se comportă ca un simplu caracter de text.

Square – imaginea se va afla în interiorul unui pătrat imaginar, care este "învelit" de text.

Tight – textul va urma conturul imaginii, care nu este neapărat doar un dreptunghi. Astfel, dacă imaginea este triunghiulară, ea se va afla în interiorul unui triunghi imaginar, "învelit" de text.

Through – prin text. Imaginea va întrerupe "curgerea textului". În acest caz, la citirea fiecăruia dintre rânduri, privirea cititorului va trebui să "sară" peste imagine.

Top and bottom – textul va "înveli" imaginea doar din părțile de sus şi de jos, fără a utiliza spațiile libere din stânga şi din dreapta acesteia.

Behind text – imaginea se va afla în spatele textului, însuşi textul suprapunându-se pe ea.

In front of text – imaginea se va afla în fața textului, acesta de sub ea fiind imposibil de citit.

În rubrica **Wrap to** (Înveleşte din) se indică părțile imaginii care vor fi înconjurate de text: ambele părți (**Both sides**), partea stângă (**Left only**), partea dreaptă (**Right only**) sau partea cea mai mare (**Largest only**). Distanța dintre text şi imagine se stabileşte în cele patru contoare ale rubricii **Distance from text**.

Dimensiunile imaginii pot fi modificate cu ajutorul paginii **Size** (Dimensiuni), iar culoarea fundalului şi a chenarului – cu ajutorul paginii **Colors and Lines** (Culori şi linii) a ferestrelor respective de dialog.

Operațiile ce pot fi efectuate asupra unei imagini sunt:
– definire (poziție şi dimensiune);
– redimensionare;
– formatarea chenarului;
– copiere;
– mutare;
– ştergere;
– ancorare.

Aplicația **Word** oferă mai multe facilități pentru realizarea acestor operații:
– opțiunile meniurilor, **Insert**, **Format** şi ale meniurilor contextuale;
– butoanele barei de instrumente standard;
– elementele de control ale ferestrelor **Format Picture**, **Format Shape**, **SmartArt Tools**, **Format Object**;
– meniurile contextuale afişate la selectarea imaginilor prin clicuri de dreapta;
– tehnici de lucru cu şoricelul.

Menționăm că utilizarea şoricelului permite efectuarea multor operații într-un mod simplu şi intuitiv. De exemplu, redimensionarea imaginii selectate se efectuează "trăgând" marcajul "↔" în direcția dorită. Poziționarea imaginii poate fi făcută utilizând tehnica "trage-şi-lasă".

În procesul inserării şi formatării imaginilor se vor respecta următoarele reguli:

1. Evitați utilizarea excesivă a imaginilor. Inserați imagini numai atunci când acestea vor adăuga un înțeles documentului în curs de elaborare. Nu uitați: imaginile inutile distrag atenția cititorului.

2. Dimensiunea şi poziţia imaginilor sunt la fel de importante ca şi conţinutul lor. O imagine de dimensiuni mari este adesea preferabilă unor numeroase imagini de dimensiuni mici.

3. Evitaţi plasarea împrăştiată pe pagină a imaginilor. Aliniaţi imaginile cu marginile paragrafelor sau unele cu altele. O imagine mică, repetată în aceeaşi poziţie pe toate paginile, oferă documentului un aspect unitar.

Întrebări şi exerciţii

❶ Cum credeţi, care este rolul imaginilor din componenţa unui document?

❷ Din care surse pot fi preluate imaginile ce vor fi inserate într-un document?

❸ Care sunt proprietăţile unei imagini? Cum pot fi afişate pe ecran aceste proprietăţi?

❹ Explicaţi termenul *formatarea imaginilor*. Ce operaţii pot fi efectuate asupra imaginilor?

❺ **EXPERIMENTEAZĂ!** Cum se comportă imaginile fără o poziţie fixă pe pagină în cazul modificării textului?

❻ E cunoscut faptul că imaginile cu o poziţie fixă pe pagină pot fi ancorate sau neancorate. Cum se comportă aceste imagini în cazul unei modificări serioase a textului?

❼ **STUDIU DE CAZ!** Afişaţi pe ecran fereastra de dialog în care sunt arătate modurile de aranjare a textului în jurul imaginilor. Încercaţi să găsiţi în acest manual imagini, textul din jurul cărora este aranjat conform schiţelor din fereastra respectivă.

❽ Numiţi facilităţile aplicaţiei **Word** destinate inserării şi formatării imaginilor. Cum credeţi, care dintre aceste facilităţi sunt mai intuitive?

❾ Inserarea şi formatarea imaginilor presupune respectarea anumitor reguli. Care sunt aceste reguli?

❿ **STUDIU DE CAZ!** Sunt oare respectate regulile de inserare şi formatare a imaginilor în figurile din acest manual? Argumentaţi răspunsul dvs.

⓫ **CREEAZĂ!** Inseraţi în documentele create anterior imagini ce ar reda mesajul operelor respective. De exemplu, textul **Glossă** poate fi completat cu imaginea unei clepsidre.

⓬ **CREEAZĂ!** Utilizând instrumentele de desenare ale aplicaţiei **Word**, creaţi documentele ce urmează:

⓭ **EXPERIMENTEAZĂ!** Utilizând sistemul de asistenţă al aplicaţiei **Word** şi un motor de căutare pe Internet, aflaţi destinaţia tuturor comenzilor aplicaţiei **Word** pentru formatarea imaginilor. Aplicaţi fiecare dintre aceste instrumente de formatare asupra imaginilor din documentele pe care le elaboraţi şi observaţi efectele acţiunilor respective.

⓮ **ÎNVAŢĂ SĂ ÎNVEŢI!** Casetele de text (*text box*) reprezintă o zonă a paginii, dimensiunea şi poziţia cărora sunt stabilite de utilizator. Pentru obiectele din interiorul casetei (texte, tabele, imagini etc.) pot fi stabilite caracteristici proprii de procesare.
Afişaţi pe ecran proprietăţile casetelor de text. Determinaţi ce operaţii pot fi efectuate asupra casetelor. Cum credeţi, în care cazuri este justificată utilizarea casetelor?

1.8. Grafica orientată pe obiecte

Termeni-cheie:
- grafică orientată pe puncte
- grafică orientată pe obiecte

E cunoscut faptul că pentru a prelucra o imagine cu ajutorul calculatorului, mai întâi ea este împărțită în microzone, numite **puncte** sau **pixeli**. Fiecare punct este reprezentat în memoria calculatorului prin unul (imagini monocolor) sau trei cuvinte binare (imagini color), iar prelucrarea imaginii se realizează prin modificarea cuvintelor respective.

Reprezentarea şi procesarea imaginilor prin împărțirea lor în microzone se numeşte *grafică orientată pe puncte*.

Grafica orientată pe puncte este folosită în aplicația **Paint**, care păstrează imaginile în fişiere cu extensia **.bmp** (*bit map* "hartă din biți"). Evident, pentru a redimensiona o imagine, se măresc sau se micşorează toate microzonele din componența acesteia. Neajunsul principal al graficii orientate pe puncte constă în faptul că redimensionarea imaginilor înrăutățește calitatea acestora. Pentru exemplificare, în *figura 1.22, a* sunt prezentate imagini redimensionate, create cu ajutorul aplicației **Paint**.

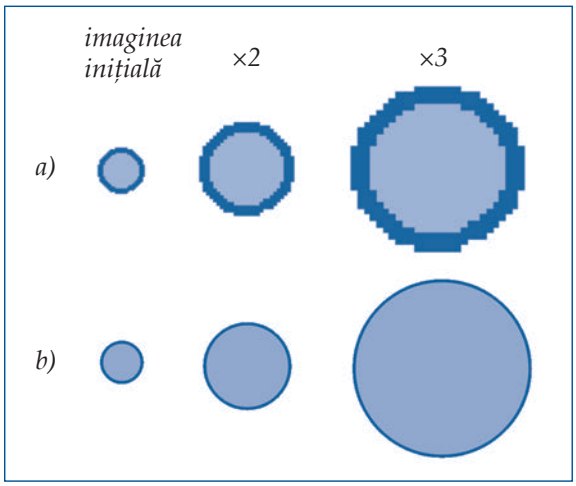


Fig. 1.22. Redimensionarea imaginilor:
a – grafica orientată pe puncte; *b* – grafica orientată pe obiecte

O calitate mai bună a imaginilor este asigurată în grafica orientată pe obiecte. În această metodă imaginile complexe sunt formate din obiecte grafice mai simple: linii, pătrate, dreptunghiuri, circumferințe, elipse etc.

Reprezentarea şi procesarea imaginilor prin împărțirea lor în obiecte grafice mai simple se numeşte *grafică orientată pe obiecte*.

În calculator imaginea realizată prin metoda graficii orientate pe obiecte este codificată printr-o listă. Fiecare element al listei conține toate informațiile necesare

pentru desenarea obiectului respectiv: coordonatele centrului şi raza fiecărui cerc, coordonatele vârfului fiecărui dreptunghi etc. Prelucrarea imaginilor se realizează prin recalcularea coordonatelor şi dimensiunilor fiecărui obiect grafic din listă. Pentru a afişa pe ecran sau a tipări o imagine, calculatorul parcurge lista respectivă şi "desenează" fiecare obiect grafic. Desenarea se realizează prin stabilirea luminanței şi culorii fiecărei microzone a ecranului sau a imprimantei. Întrucât dimensiunile microzonelor nu mai depind de dimensiunile obiectelor grafice, calitatea imaginilor nu se schimbă la redimensionarea acestora (*fig. 1.22, b*).

Pentru grafica orientată pe obiecte, aplicația **Word** conține meniul dedicat **Shapes**. Accesul la facilitățile oferite de acest meniu se realizează prin intermediul ferestrei de dialog, care este afişată imediat după lansarea comenzii **Insert**, **Shapes** (*fig. 1.23*).

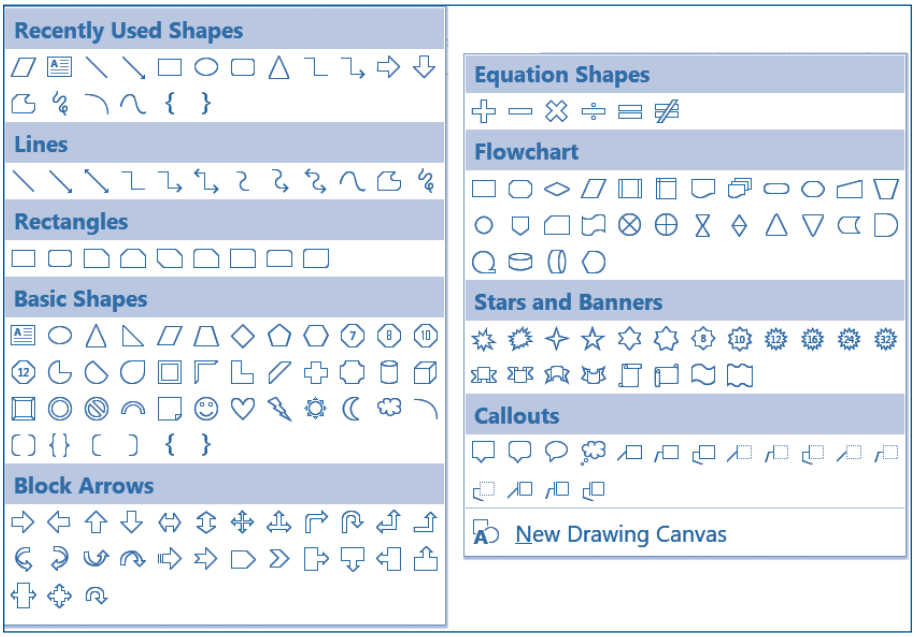


Fig. 1.23. Instrumente în grafica orientată pe obiecte

Instrumentele din figura de mai sus permit desenarea următoarelor obiecte grafice:
– linii;
– pătrate şi dreptunghiuri;
– cercuri şi elipse;
– arce de cerc sau de elipse;
– forme neregulate;
– casete explicative;
– figuri predesenate etc.

Obiectul grafic dorit se selectează prin acționarea butoanelor respective. Accentuăm faptul că fiecare obiect grafic va ocupa o poziție fixă pe pagină, indiferent de conținutul acesteia.

Pentru a crea o imagine complexă, utilizatorul inserează consecutiv obiectele dorite: linii, pătrate, dreptunghiuri, cercuri etc. Imediat după inserare se stabilesc

proprietățile fiecărui obiect: grosimea şi culoarea liniei de contur, culoarea de umplere, efectele speciale. În procesul inserării, obiectele grafice pot fi suprapuse unul peste altul. **Ordinea suprapunerilor** poate fi schimbată cu ajutorul comenzilor **Bring to Front** şi **Sent to Back**, care permit mutarea în fața sau în spatele obiectului dorit (*fig. 1.24*).

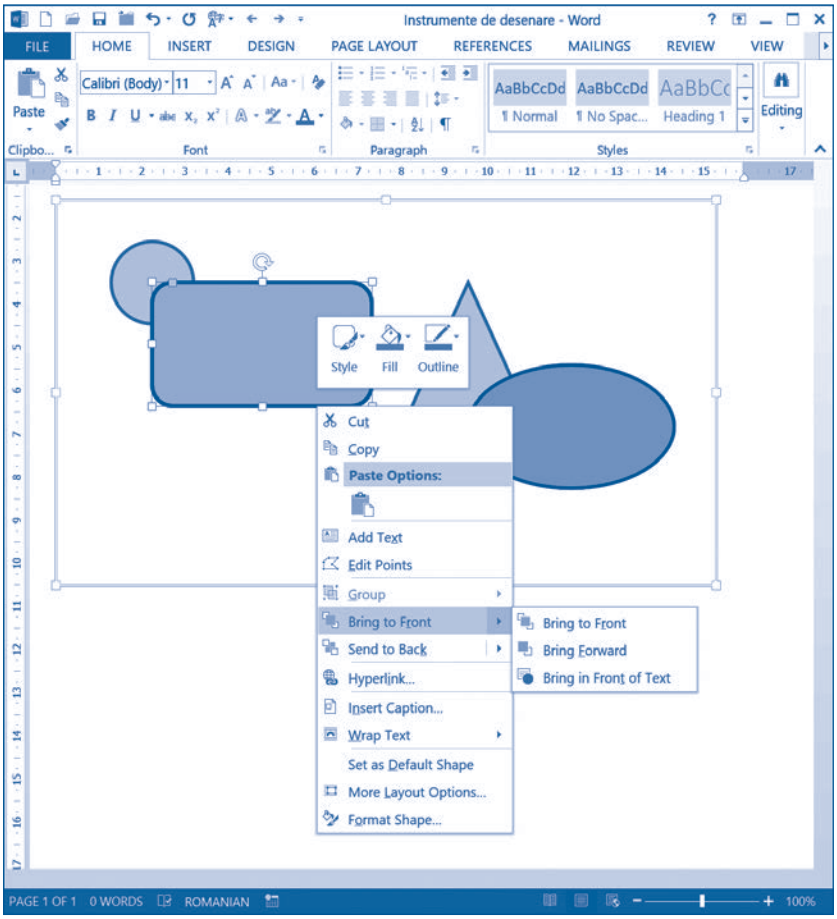


Fig. 1.24. Comanda **Bring to Front**

O altă operație frecvent utilizată în crearea imaginilor este **gruparea obiectelor**. Prin grupare, două sau mai multe obiecte grafice sunt tratate ca un singur obiect. După efectuarea acestei operații, obiectele grafice care formează un singur grup pot fi:

– mărite sau micşorate la aceeaşi scară;
– mutate împreună, păstrându-şi poziția relativă unul față de celălalt;
– şterse;
– copiate dintr-un document în altul etc.

În caz de necesitate, utilizatorul poate dezmembra grupa, fiecare obiect grafic devenind independent de celelalte.

Meniul contextual **Format Shape** (*fig. 1.25*) oferă şi alte opțiuni pentru procesarea obiectelor grafice:

– umbrirea (**Shadow**);
– reflectarea (**Reflection**);
– aureola (**Glow**);
– formatarea tridimensională (**3-D Format**);
– rotirea obiectului grafic selectat (**Rotate**);
– oglindirea față de verticală sau orizontală (**Flip**);
– alinierea obiectelor grafice (**Align**).

Alinierea poate fi făcută la o rețea de linii orizontale sau verticale, denumite sugestiv *canava*, la extremitățile paginii fizice sau la alte obiecte grafice.

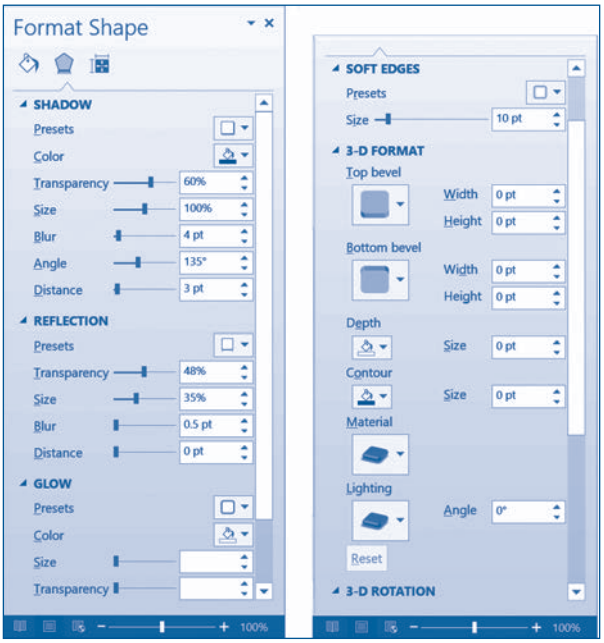


Fig. 1.25. Fereastra de dialog **Format Shape**

Casetele explicative (**Callouts**) reprezintă un tip special de obiecte grafice, folosite la explicarea unor elemente din document (*fig. 1.26*).

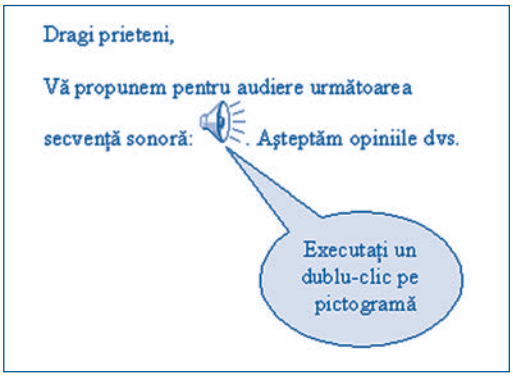


Fig. 1.26. Casetă explicativă

Casetele explicative sunt alcătuite dintr-o casetă de text şi o legătură cu elementul explicat. Pentru textul din interiorul casetei pot fi stabilite caracteristici proprii de formatare. Utilizatorul poate insera în document casete explicative de diferite forme. Configurația acestora poate fi modificată cu ajutorul şoricelului. Formatarea textului din casetă se efectuează cu ajutorul comenzii **Format** sau al butoanelor respective din bara de instrumente de formatare.

Întrebări şi exerciții

❶ Explicați termenul *grafică orientată pe puncte*. Cum se codifică și se prelucrează imaginile în grafica orientată pe puncte?

❷ **EXPERIMENTEAZĂ!** Creați în aplicația **Paint** imaginea inițială din *figura 1.22, a*. Măriți imaginea creată de 2, 4 și de 6 ori. Observați cum se schimbă calitatea imaginii. Explicați cauza acestor schimbări.

❸ Explicați termenul *grafică orientată pe obiecte*. Cum se codifică și se prelucrează imaginile în grafica orientată pe obiecte?

❹ **EXPERIMENTEAZĂ!** Creați în aplicația **Word** imaginea inițială din *figura 1.22, b*. Măriți imaginea creată de 2, 4 și de 6 ori. Se schimbă oare calitatea imaginii în procesul redimensionării?

❺ Ce operații efectuează calculatorul în procesul afișării unei imagini realizate în grafica orientată pe obiecte?

❻ **EXPLOREAZĂ!** Găsiți în meniuri și în ferestrele de dialog obiectele grafice predesenate, oferite de aplicația **Word** pentru crearea imaginilor.

❼ **ÎNVAȚĂ SĂ ÎNVEȚI!** Utilizând sistemul de asistență, aflați destinația tuturor butoanelor din meniurile și ferestrele de dialog cu instrumente de desenare.

❽ Vizualizați toate figurile predesenate oferite de aplicația **Word**. Cum credeți, în ce fel de documente ar putea fi folosite aceste obiecte grafice?

❾ Care este destinația operației de grupare? Când se utilizează această operație?

❿ Cum poate fi schimbată ordinea de suprapunere a obiectelor grafice? Când apare necesitatea de a schimba ordinea de suprapunere?

⓫ **ÎNVAȚĂ SĂ ÎNVEȚI!** Utilizând sistemul de asistență, aflați destinația tuturor opțiunilor din fereastra de dialog **Format Shape**. Verificați cum se modifică obiectele grafice la activarea acestor opțiuni.

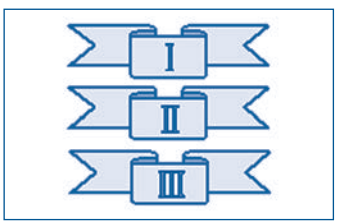
⓬ Când se utilizează casetele explicative? Cum pot fi formatate aceste casete?

⓭ **STUDIU DE CAZ!** Schițați câteva imagini ce pot fi create atât în aplicația **Paint** (grafică orientată pe puncte), cât și în aplicația **Word** (grafică orientată pe obiecte). Creați aceste imagini în ambele aplicații. Comparați calitatea imaginilor obținute în cazul redimensionării acestora.

⓮ **CREEAZĂ!** Realizați cu ajutorul aplicației **Word** imaginile din *figura 1.27*.

a)

b)



c) d)

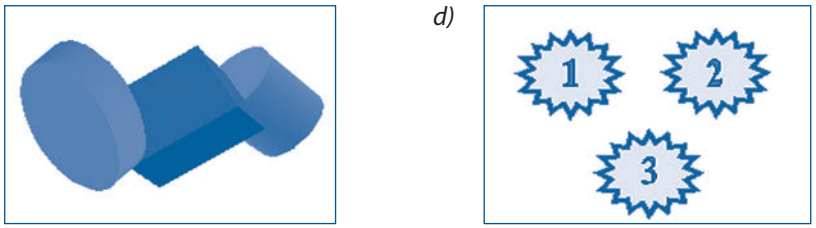


Fig. 1.27. Desene create în grafica orientată pe obiecte

1.9. Diagrame

Termeni-cheie:
- diagramă
- foaie de date
- formatare a diagramei
- tip de diagramă

Majoritatea cititorilor sunt intimidați de textele care conțin foarte multe numere. Datele numerice sunt mult mai sugestive atunci când sunt prezentate într-o formă grafică.

***Diagrama* reprezintă o imagine în care valorile datelor numerice sunt redate prin dimensiunile unor obiecte grafice.**

O diagramă este formată din următoarele obiecte (*fig. 1.28*): titlul (opțional); zona de desenare; axa categoriilor, de obicei axa *x*; axa valorilor, de obicei axa *y*; indicatorii de date; legenda.

Valorile numerice sunt redate prin dimensiunile indicatorilor de date: lungime, lățime, înălțime etc.

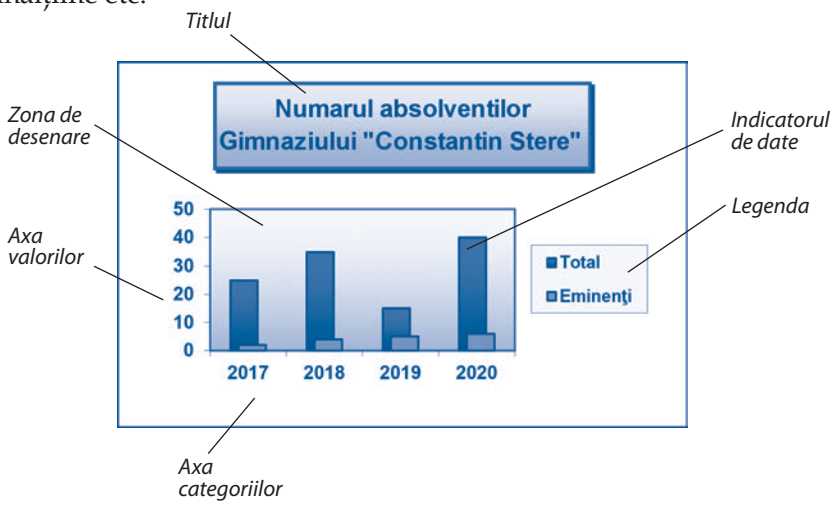


Fig. 1.28. Structura diagramelor

De exemplu, axa categoriilor din *figura 1.28* conține categoriile 2017, 2018, 2019, 2020 sau, cu alte cuvinte, anii pentru care se indică numărul de absolvenți. Axa valorilor servește pentru determinarea numărului de absolvenți și conține o scară formată din numerele 0, 10, 20, …, 50. Indicatorii de date sunt în formă de dreptunghiuri (coloane), înălțimea lor indicând numărul de absolvenți din anii respectivi.

Menționăm faptul că în diagrama din *figura 1.28* sunt redate două serii de date, prima fiind numărul total de absolvenți, iar a doua – numărul de eminenți din fiecare promoție. Pentru a reprezenta valorile din fiecare serie, se folosesc indicatori de date colorați în mod diferit. Semnificația culorilor este explicată în legendă.

Diagramele se creează cu ajutorul aplicației **Microsoft Graph Chart** (Diagrame Microsoft). În această aplicație, pentru fiecare diagramă se completează un șablon special, numit foaie de date (*fig. 1.29*).

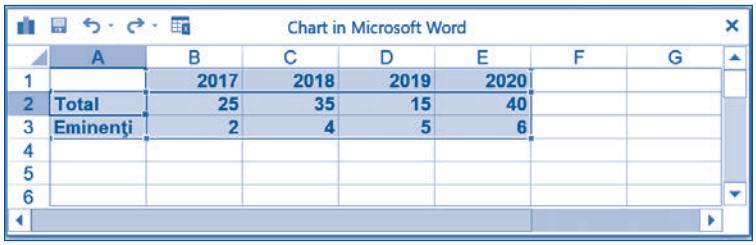


Fig. 1.29. Foaie de date

***Foaia de date* reprezintă un tabel a cărui informație este utilizată pentru crearea diagramelor.**

Primul rând și prima coloană ale unei foi de date sunt rezervate pentru textul care identifică informația din celulele respective. Celelalte rânduri și coloane sunt destinate pentru introducerea valorilor numerice, fiind notate prin numere și litere.

De exemplu, primul rând din *figura 1.29* conține categoriile 2017, 2018, 2019 și 2020. Prima coloană a aceleiași foi de date conține denumirile seriilor de date – Total și Eminenți. Celula aflată la intersecția rândului 2 și a coloanei B conține numărul de absolvenți din anul 2017, egal cu 25; celula din rândul 3, coloana C, conține numărul de eminenți din promoția anului 2018, egal cu 4 etc.

Asupra foilor de date pot fi efectuate următoarele operații:
– modificarea, ștergerea sau copierea datelor din celule;
– excluderea sau includerea rândurilor și coloanelor;
– redimensionarea coloanelor.

Majoritatea comenzilor care realizează aceste operații sunt incluse în meniurile contextuale, care pot fi afișate pe ecran poziționând cursorul pe obiectul dorit. Operațiile frecvent utilizate pot fi efectuate cu ajutorul butoanelor din ferestrele de dialog ce sunt afișate la lansarea comenzilor respective.

Fiecare obiect din componența unei diagrame se caracterizează prin anumite proprietăți și este relativ independent de celelalte elemente. De exemplu, **Titlul** se caracterizează prin formatările de text, culoare și fundal. **Axele** se caracterizează prin stilul liniei, scară, formatările de numere, orientarea și alinierea textului. **Indicatorii de date** se caracterizează prin formă, stilul de marcare, culoare etc. Indicatorii identici reprezintă o serie de date.

Numim *formatare a diagramei* procesul de stabilire a proprietăților obiectelor din componența ei: titlul, zona de desenare, axele, legenda, indicatorii de date.

Pentru a efectua o operație de formatare, mai întâi se selectează obiectul dorit. Selectarea obiectului impune apariția pe ecran în **Format Data Series** (Formatarea seriilor de date) a unei ferestre de dialog care conține elementele de control destinate stabilirii tuturor proprietăților obiectului selectat (*fig. 1.30*).

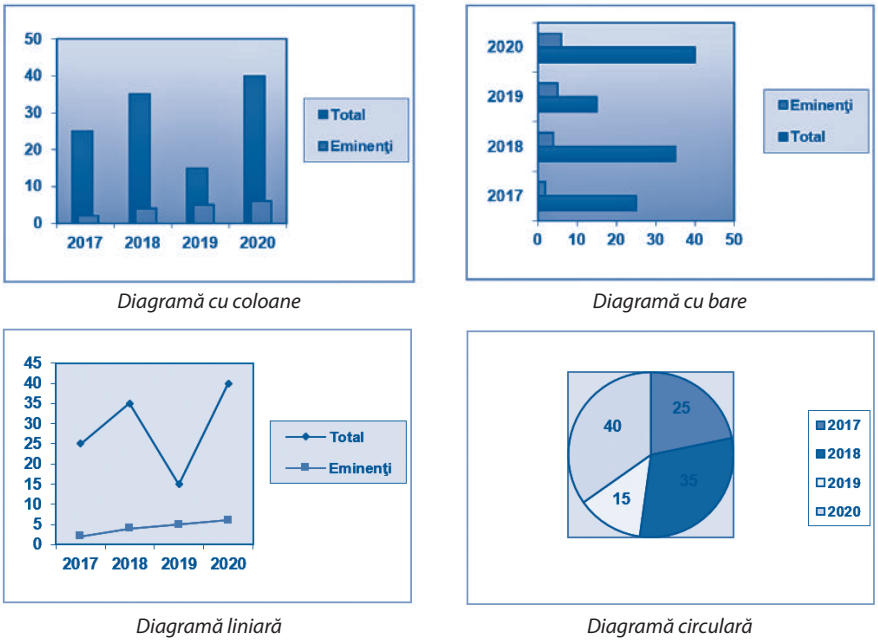
Fig. 1.30. Fereastra de dialog **Format Data Series**

Aplicația **Microsoft Graph Word** oferă mai multe tipuri de diagrame. Tipul de diagramă se defineşte conform obiectului grafic folosit pentru reprezentarea valorilor numerice: coloane, bare, linii, sectoare de cerc etc. Cele mai frecvent utilizate tipuri de diagrame sunt (*fig. 1.31*, p. 44):

– diagrame cu coloane;
– diagrame cu bare;
– diagrame liniare;
– diagrame circulare.

Evident, fiecare tip de diagramă afişează datele în mod diferit. Alegerea tipului de diagramă se face conform recomandărilor ce urmează:

1. Diagramele cu coloane se utilizează pentru a reprezenta diferite serii de date care se schimbă în timp. Pentru exemplificare amintim numărul de absolvenți din fiecare promoție, veniturile lunare ale părinților, vânzările unei societăți comerciale etc.



Diagramă cu coloane *Diagramă cu bare*

Diagramă liniară *Diagramă circulară*

Fig. 1.31. Tipuri de diagrame

2. Diagramele cu bare se aplică pentru a compara serii de date care nu se schimbă în timp. De exemplu, o astfel de diagramă poate fi folosită pentru a reprezenta capacitățile de prelucrare ale diferitor calculatoare.

3. Diagramele liniare sunt cele mai potrivite pentru a reprezenta o tendință sau o relație dintre anumite valori pe o perioadă de timp. Drept exemplu, amintim temperatura unui pacient din spital, cursul leului etc.

4. Diagramele circulare se folosesc pentru a evidenția raportul dintre părți și întreg. De exemplu, în cazul unei rețete culinare fiecare sector de cerc reprezintă cantitatea unui produs dintr-un anumit fel de mâncare.

Pentru a stabili tipul de diagramă, se activează comanda **Change Chart Type** (Schimbă tipul diagramei).

Accentuăm faptul că meniurile **Design** și **Format** din grupul **Chart Tools** conțin mai multe opțiuni pentru diagrame, fapt ce permite poziționarea legendei, indicarea pe diagramă a anumitor valori, stabilirea dimensiunilor de axe, afișarea grilei etc.

Întrebări și exerciții

❶ **CERCETEAZĂ!** Care este destinația diagramelor? Găsiți în manualele de istorie și de geografie câte trei diagrame cel puțin.

❷ Numiți obiectele din componența unei diagrame. Ce proprietăți au aceste obiecte?

❸ Explicați structura unei foi de date. Ce operații pot fi efectuate asupra foilor de date?

❹ Care este legătura dintre o foaie de date și diagrama corespunzătoare?

❺ Explicați termenul *formatarea diagramelor*. Cum se efectuează operațiile de formatare?

❻ **STUDIU DE CAZ!** Numiți tipurile de diagrame frecvent utilizate. Când se utilizează fiecare tip de diagramă? Dați exemple.

❼ **CERCETEAZĂ!** Determinați tipul diagramelor din manualele de istorie și de geografie.

❽ **ELABOREAZĂ!** Creați o diagramă similară celei din *figura 1.28* (p. 41). Introduceți în foaia de date numărul de absolvenți ai gimnaziului dvs.

❾ **CREEAZĂ!** Folosind datele din tabelul ce urmează, creați diagrama cu coloane *Venitul companiei*.

	Ianuarie	Februarie	Martie	Aprilie	Mai	Iunie
Secția A	70	50	90	20	100	40
Secția B	100	150	110	180	30	120

❿ **LUCREAZĂ ÎN ECHIPĂ!** Creați o diagramă cu bare ce reprezintă notele medii ale prietenilor dvs. Pe axa *x* se afișează notele, iar pe axa *y* se indică numele și prenumele fiecărui elev.

⓫ Folosind datele din tabelul ce urmează, creați diagrama liniară *Cursul dolarului american în raport cu leul moldovenesc*.

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Lei	12,30	12,15	11,72	12,06	13,05	15,62	19,66	19,98	17,10	17,14	17,21

⓬ **CREEAZĂ!** Parcul auto al unei întreprinderi este compus din 60 de autoturisme, 20 de camioane și 15 autobuze. Reprezentați componența parcului auto cu ajutorul unei diagrame circulare.

⓭ **ÎNVAȚĂ SĂ ÎNVEȚI!** În aplicația **Word**, diagramele pot fi create și în baza tabelelor deja existente în document. Pentru aceasta se selectează datele din tabel și se copie în foaia de date a diagramei în curs de construcție. Inserați într-un document tabelele din exercițiile 9 și 11. Creați pe baza acestor tabele diagrame cu coloane, diagrame cu bare și diagrame circulare. Cum credeți, care tip de diagramă este mai sugestiv?

1.10. Verificarea textelor

Termeni-cheie:
- analizor lexical
- analizor gramatical
- tezaur

În procesul elaborării și tastării documentelor pot apărea erori. Aplicația **Word** dispune de programe speciale, destinate verificării textelor dintr-un document. Evident, pentru a verifica un text, mai întâi trebuie cunoscută limba în care el a fost scris. Întrucât alfabetul latin este folosit de mai multe limbi, calculatorul nu poate determina univoc apartenența lingvistică a textului, acest lucru revenind utilizatorului.

Informația referitoare la limba în care este scris un fragment de text (română, engleză, franceză etc.) se include în proprietățile de caracter sau de paragraf cu ajutorul opțiunii **Language**, **Set Proofing Language** (Limba, Definire limbă) din meniul

Review (Revizuire). La activarea acestei comenzi pe ecran este afişată fereastra de dialog din *figura 1.32*.

Lista derulantă a ferestrei examinate include limbile pentru care sunt disponibile programe de verificare automată a textului. Evident, programele respective trebuie să fie instalate pe calculator. După indicarea limbii dorite, aplicaţia memorează în formatările de caracter apartenenţa lingvistică a fragmentului selectat de text.

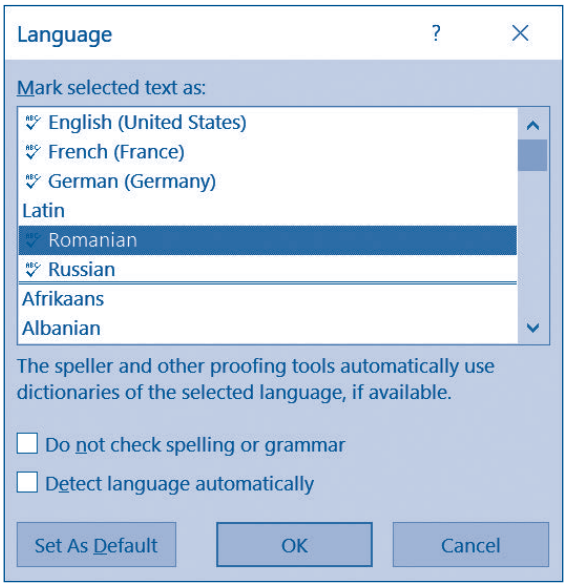


Fig. 1.32. Fereastra de dialog **Language**

I *Analizorul lexical* este un program care verifică scrierea corectă a fiecărui cuvânt.

Analizorul lexical confruntă (compară) cuvintele din textul supus verificării cu cele dintr-o listă de cuvinte scrise corect, numită **dicţionar**. Dacă cuvântul din text nu este găsit în dicţionar, se semnalează o eroare.

I *Analizorul gramatical* este un program care verifică scrierea corectă a fiecărei propoziţii.

Funcţionarea analizorului gramatical se bazează pe un set de reguli cu privire la modificarea formelor cuvintelor şi îmbinarea lor în propoziţii.

Pentru a verifica textul unui document, se activează opţiunea **Spelling and Grammar** (Analizor lexical şi Analizor gramatical) din meniul **Review** sau butonul cu acelaşi nume. Programele de verificare parcurg textul şi, în caz de eroare, afişează o fereastră de dialog în care se indică tipul greşelii. De exemplu, în cazul unei greşeli lexicale, se afişează fereastra de dialog din *figura 1.33*.

Fragmentul de text care, din punctul de vedere al calculatorului, conţine o greşeală este afişat în zona **Not in Dictionary** (Nu-i în dicţionar). Cuvântul greşit sau, mai exact, cuvântul ce nu a fost găsit în dicţionar este evidenţiat cu o linie ondulată de culoare roşie. Utilizatorul poate corecta greşeala direct în fereastra de dialog, efectuând tastările respective sau alegând unul dintre cuvintele corecte propuse de calculator în zona **Suggestions** (Sugestii).

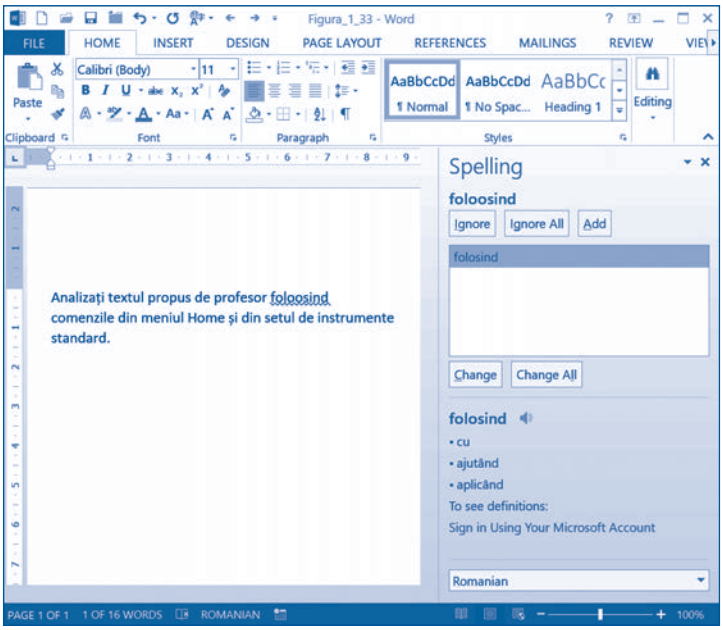


Fig. 1.33 Semnalarea erorilor lexicale

Elementele de control ale unei ferestre **Spelling** au următoarea semnificație:

Ignore (Ignoră) – la acționarea acestui buton se va trece la verificarea textului ce urmează după cuvântul evidențiat. Se utilizează în cazul unor cuvinte corecte care nu au fost incluse în dicționar, de exemplu, denumirile unor sate sau orașe.

Ignore All (Ignoră toate aparițiile) – în cazul următoarelor apariții ale cuvântului evidențiat nu vor mai fi semnalate erori.

Add (Include) – cuvântul evidențiat va fi inclus în dicționar. În continuare astfel de cuvinte vor fi tratate ca fiind corecte. Se utilizează pentru includerea în dicționar a unor cuvinte folosite în activitatea profesională, de exemplu a unor termeni medicali.

Change (Modifică) – acest buton se acționează după corectarea textului din fereastra de dialog. Modificările făcute vor fi copiate în textul supus verificării.

Change All (Modifică toate aparițiile) – aparițiile ulterioare ale cuvântului evidențiat vor fi corectate în mod automat.

După verificarea tuturor cuvintelor din propoziția curentă, analizorul gramatical controlează propoziția în ansamblu. În caz de nerespectare a regulilor de modificare şi îmbinare a cuvintelor în propoziții, se afișează o fereastră de dialog în care se indică tipul erorii. De exemplu, fereastra de dialog din *figura 1.34* (p. 48) semnalează eroarea gramaticală **Repeated Word** (Cuvânt repetat).

În procesul verificării, analizorul gramatical calculează următoarele **date statistice** (*fig. 1.35*, p. 48):

– numărul de pagini;
– numărul de cuvinte;
– numărul de caractere, fără spații;
– numărul de caractere, inclusiv spațiile;
– numărul de paragrafe;
– numărul de rânduri.

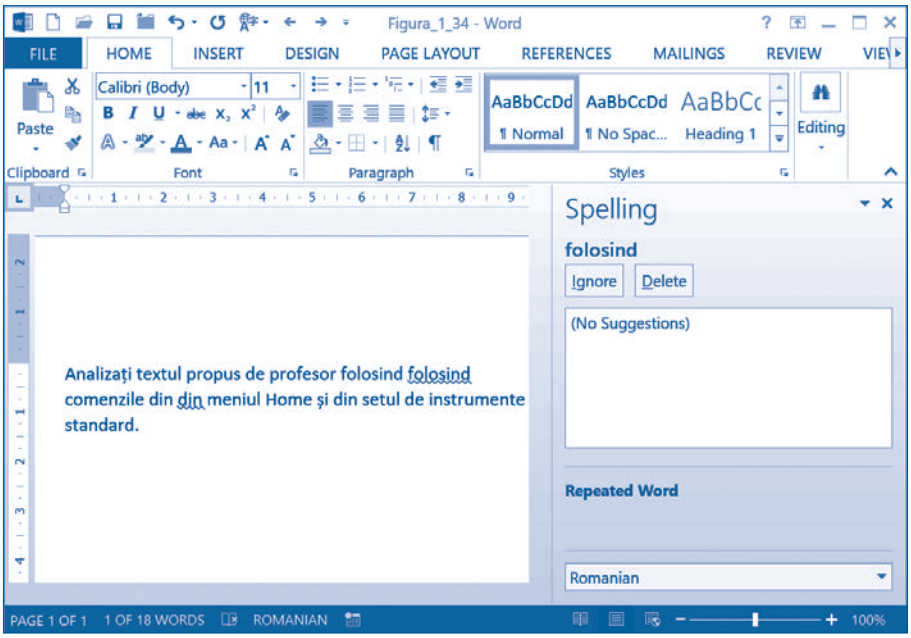


Fig. 1.34. Semnalarea erorilor gramaticale

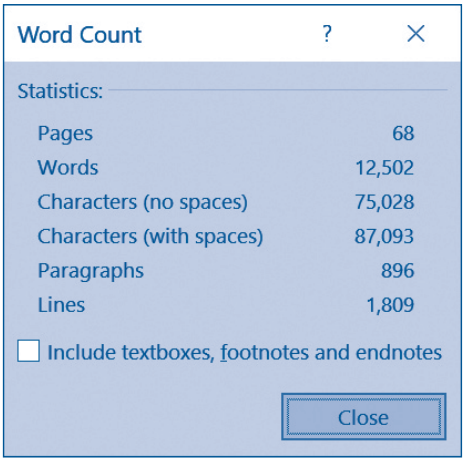


Fig. 1.35. Date statistice despre text

Aceste date se afişează cu ajutorul comenzii **Word Count** (Numărarea cuvintelor) din meniul **Spelling and Grammar** (Ortografie şi gramatică) din panglica de meniuri **Review**. Ele pot fi de folos atunci când se doreşte adaptarea textului la nivelul de pregătire a cititorului. De exemplu, în cazul unei reclame se folosesc propoziții scurte, care pot fi citite şi înțelese rapid. De asemenea, în cazul creării unor documente originale, de exemplu a unor opere artistice, datele respective sunt utilizate pentru a calcula remunerarea autorilor.

Calitatea unui text fără greşeli gramaticale poate fi îmbunătățită prin excluderea repetărilor enervante şi plictisitoare. Dacă unul şi acelaşi cuvânt se întâlneşte prea

des într-un paragraf, se recomandă înlocuirea lui cu sinonime sau cuvinte înrudite. În acest scop, se foloseşte programul **Thesaurus** (Tezaur), care poate fi apelat prin lansarea comenzii cu acelaşi nume din meniul **Proofing** (Verificare).

> ***Tezaurul* este un program care înlocuieşte cuvintele selectate cu sinonime sau cuvinte înrudite.**

Pentru a înlocui un cuvânt, se activează comanda **Thesaurus** din meniul **Proofing**. Fereastra de dialog **Thesaurus** (*fig. 1.36*) conţine cuvântul selectat (**Looked Up**), semnificaţia lui (**Meanings**) şi o listă de sinonime (**Replace with Synonym**).

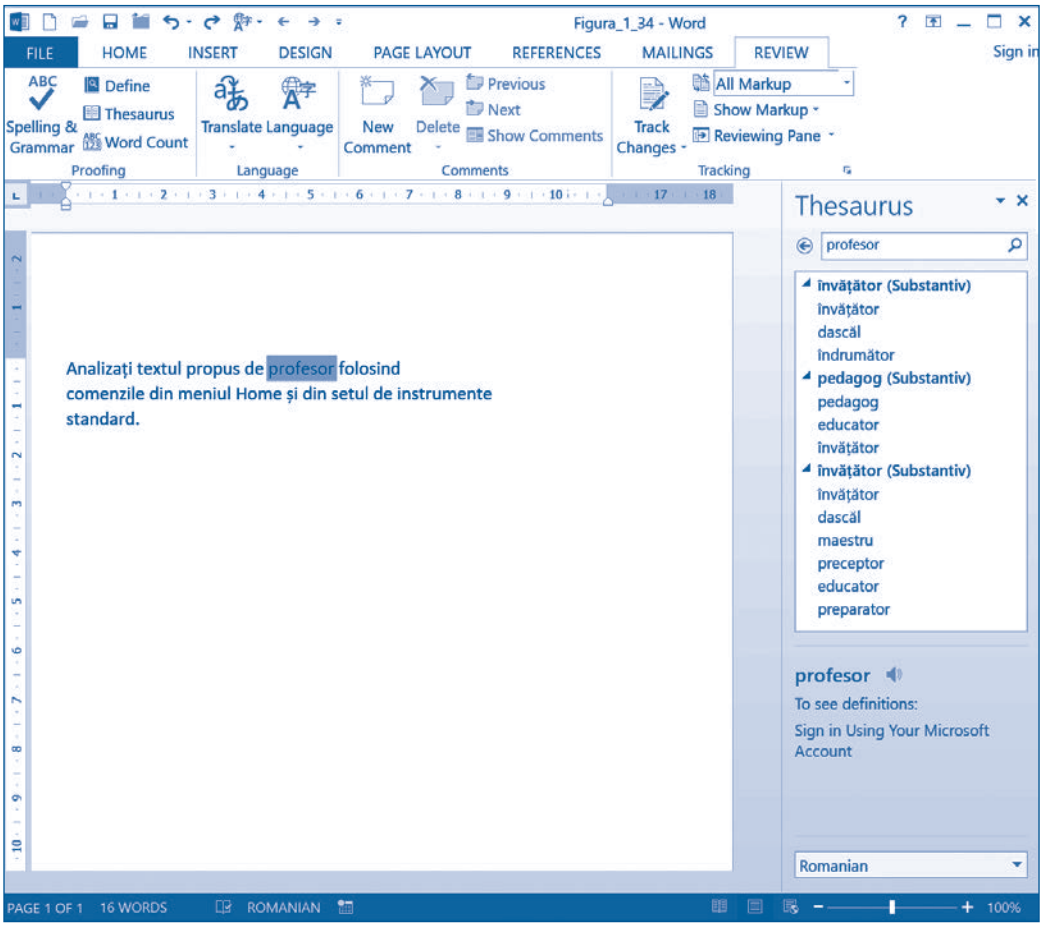


Fig. 1.36. Fereastra de dialog **Thesaurus**

Pentru a înlocui cuvântul selectat în text, se acţionează butonul **Insert** (Inserează), iar pentru a-l copia în memoria-tampon – butonul **Copy**.

Menţionăm faptul că regulile gramaticale, relativ uşor înţelese de oameni, se transpun foarte greu într-un limbaj înţeles de calculator. Calculatorul nu poate "ghici" sensul textului, mai ales atunci când acesta conţine greşeli. În consecinţă, sugestiile propuse de programele de verificare uneori sunt nepotrivite. Prin urmare, existenţa unor instrumente de verificare a textelor nu scuteşte utilizatorul de

necesitatea cunoaşterii profunde a limbii române şi a limbilor de circulaţie internaţională.

În cazul limbilor de circulaţie internaţională, procesoarele moderne de texte oferă utilizatorului posibilitatea de a face traduceri automate. Comanda **Translate** este localizată în meniul **Language** al grupului de meniuri **Review**. Traducerea propriu-zisă este efectuată fie de un program local, care, evident, trebuie să fie instalat pe calculatorul utilizatorului, fie de un serviciu de traduceri online.

Întrebări şi exerciţii

❶ Cum credeţi, pentru ce se indică apartenenţa lingvistică a fiecărui fragment de text?

❷ Explicaţi termenii *analizor lexical* şi *analizor gramatical*.

❸ Cum se comunică calculatorului apartenenţa lingvistică a unui fragment de text? Unde se va păstra această informaţie?

❹ Care este destinaţia dicţionarului din componenţa analizorului lexical? Ce informaţii conţine acest dicţionar?

❺ Ce operaţii efectuează calculatorul în procesul verificării lexicale a textului?

❻ Ce informaţii sunt necesare pentru funcţionarea analizorului gramatical? Ce operaţii efectuează acest program în procesul verificării?

❼ **CREEAZĂ!** Elaboraţi un document care conţine textele exerciţiilor 1, 2 şi 3. Afişaţi pe ecran apartenenţa lingvistică a fiecărui paragraf. Verificaţi textul şi salvaţi documentul în fişierul **Exerciţii**.

❽ **EXPERIMENTEAZĂ!** Inseraţi în textul documentului **Exerciţii** următoarele erori:
a) repetarea unei litere într-un cuvânt;
b) repetarea a două cuvinte într-o propoziţie;
c) schimbarea locului a două cuvinte într-o propoziţie;
d) includerea unui cuvânt arbitrar într-o propoziţie.
Lansaţi programele de verificare a textului şi explicaţi mesajele afişate pe ecran. Corectaţi textul conform sugestiilor propuse de calculator.

❾ Care este destinaţia ferestrei de dialog *Thesaurus*? În care cazuri se utilizează acest program?

❿ Corectaţi textele din fişierele propuse de profesor.

⓫ Verificaţi textele din documentele create anterior: **Cântec**, **Amintiri din copilărie**, **Peripeţiile Alicei...**, **Şi dacă...**.

⓬ Scrieţi un text în limba străină pe care o studiaţi în şcoală. Verificaţi acest text cu ajutorul calculatorului.

⓭ Stabiliţi următoarea apartenenţă lingvistică pentru paragrafele documentului **Exerciţii**: exerciţiul 1 – *româna*, exerciţiul 2 – *engleza*, exerciţiul 3 – *franceza*. Încercaţi să verificaţi acest text cu ajutorul calculatorului. Explicaţi mesajele afişate pe ecran.

⓮ **ÎNVAŢĂ SĂ ÎNVEŢI!** Utilizând sistemul de asistenţă, studiaţi de sine stătător facilităţile de traducere automată a textelor, oferite de aplicaţia **Word**. Traduceţi câteva texte pe care le-aţi studiat la lecţiile de limbi străine. Evaluaţi calitatea traducerilor făcute de aplicaţia **Word**.

1.11. Inserarea formulelor

Termeni-cheie:
- formulă
- paletă de simboluri
- paletă de şabloane

În limbajul uzual **formula** reprezintă o combinaţie de litere, cifre şi semne matematice care redau într-un mod exact o propoziţie sau o regulă pentru efectuarea unei operaţii. Pentru exemplificare, prezentăm câteva formule studiate la lecţiile de fizică şi de matematică:

– media aritmetică a n numere: $\overline{x} = \frac{x_1 + x_2 + \ldots + x_n}{n}$;

– reuniunea mulţimilor A şi B: $A \cup B = \{x \in A$ sau $x \in B\}$;

– greutatea unui corp: $G = mg$;

– constanta elastică a resortului: $k = \frac{F}{\Delta x}$;

– densitatea unui corp: $\rho = \frac{m}{V}$.

Editorul de texte **Word** oferă utilizatorului două metode de includere a formulelor în documente:

1) ca fragmente obişnuite de text;

2) ca obiecte distincte, create cu ajutorul unor comenzi dedicate.

Prima metodă se utilizează în cazul unor formule simple, literele, cifrele şi semnele matematice ale cărora pot fi găsite pe tastatură sau selectate cu ajutorul comenzii **Insert**, **Symbol** (*fig. 1.37*).

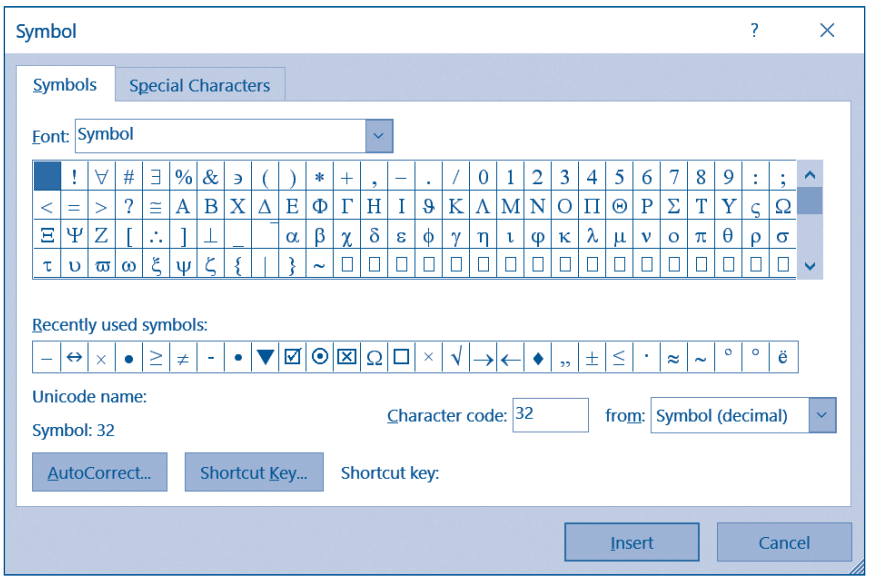


Fig. 1.37. Fereastra de dialog **Symbol**

Metoda a doua presupune utilizarea comenzii **Insert**, **Equation**. Imediat după lansarea acestei comenzi, pe ecran apare panglica de meniuri **Design** din grupul de meniuri **Equation Tools** (*fig. 1.38*).

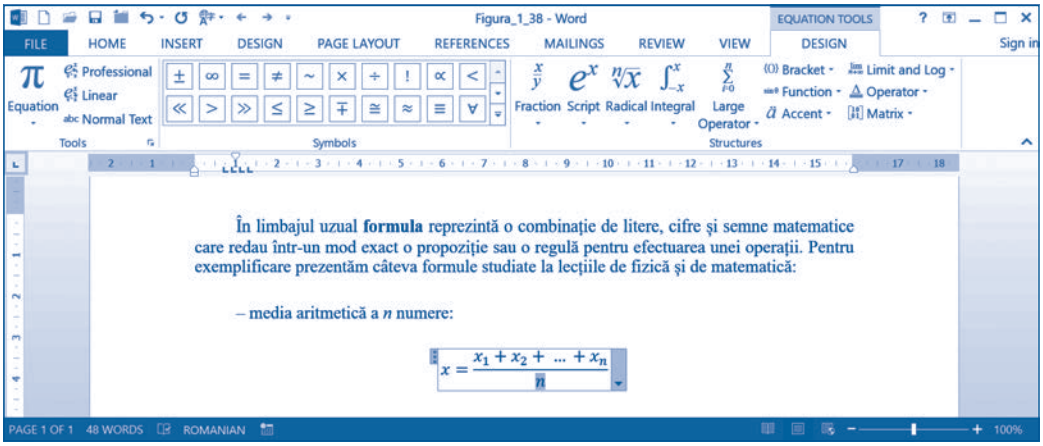


Fig. 1.38. Grupul de meniuri **Equation Tools**

Comenzile din aceste meniuri oferă accesul la cele mai diverse colecții de simboluri și șabloane matematice, denumite palete. **Paleta de simboluri** reprezintă o extensiune a tastaturii și include un set de semne matematice înrudite, de exemplu ≈, ≡, ± etc. Selectarea unui simbol din paletă are ca efect inserarea lui în formulă. **Paleta de șabloane** include un set de câmpuri destinate scrierii indiciilor, fracțiilor, radicalilor și a altor elemente frecvent întâlnite în formulele din matematică, fizică, chimie etc. Selectarea unui șablon are ca efect inserarea în formulă a câmpurilor respective care ulterior pot fi completate cu litere, cifre, semne matematice sau cu alte șabloane.

Întrebări și exerciții

❶ Explicați modalitățile de includere a formulelor în documente. Cum credeți, care sunt avantajele și dezavantajele fiecărei metode?

❷ **EXPLOREAZĂ!** Aflați cu ajutorul sistemului de asistență destinația tuturor comenzilor din aplicația **Equation Tools**. Stabiliți experimental componența seturilor de simboluri și de șabloane oferite de această aplicație.

❸ Creați documentul ce urmează:

Din tabel se observă că pentru un resort se respectă relația:

$$\frac{M_1}{\Delta L_1} = \frac{M_2}{\Delta L_2} = \frac{M_3}{\Delta L_3} = \frac{M_4}{\Delta L_4} = k,$$

unde *k* reprezintă constanta elastică a resortului.

❹ **STUDIU DE CAZ!** Creați un document care conține primele trei propoziții ale paragrafului în studiu. Utilizați pentru inserarea formulelor ambele metode – ca fragmente obținute de text și ca obiecte distincte. Cum credeți, care metodă este mai simplă?

1.12. Stiluri și șabloane

Termeni-cheie:
- stil de caractere
- stil de paragraf
- şablon

Crearea unor documente cu un aspect plăcut şi uşor de citit presupune formatarea omogenă a fragmentelor de text care au aceeaşi destinaţie. De exemplu, în acest manual astfel de fragmente sunt:

– titlul fiecărui capitol;
– titlul fiecărei teme;
– textul de bază;
– denumirile de tabele şi figuri;
– titlul *Întrebări şi exerciţii*;
– întrebările şi exerciţiile propriu-zise.

În principiu, utilizatorul poate să memoreze proprietăţile fiecărui fragment de text (fontul şi dimensiunea caracterelor, alinierea şi indentarea paragrafelor, dimensiunea paginilor etc.) şi să le aplice pentru formatarea fragmentelor similare din cuprinsul întregului document. Însă o astfel de metodă este incomodă şi ineficientă.

Aplicaţia **Word** oferă următoarele facilităţi pentru crearea şi formatarea documentelor:

– afişarea formatărilor de paragraf şi de caractere;
– preluarea formatărilor;
– formatarea cu stiluri;
– crearea şi utilizarea şabloanelor.

Formatările de paragraf şi de caractere pot fi afişate cu ajutorul elementelor de control din meniurile aplicaţiei **Word**, care îşi schimbă aspectul în funcţie de obiectul selectat. Selectând diferite obiecte ale documentului, putem afişa proprietăţile caracterelor şi paragrafelor respective.

Formatările unui fragment de text pot fi **preluate şi aplicate** altor fragmente cu ajutorul butonului **Format Painter** (Pensulă de formatare) din panglica de meniuri **Home**. La acţionarea acestui buton, aplicaţia memorează formatările fragmentului de text în care se află punctul de inserare, iar cursorul îşi schimbă forma într-o pensulă. Cu această pensulă "vopsim" textul care dorim să fie formatat exact ca fragmentul iniţial.

Formatarea cu stiluri este cea mai indicată metodă pentru prelucrarea unor documente mari. Deosebim stiluri de caractere şi stiluri de paragraf.

Numim *stil de caractere* un anumit set de proprietăţi de caracter: font, stil de afişare, dimensiune, culoare de afişare, efecte speciale.

Numim *stil de paragraf* un anumit set de proprietăţi de paragraf – aliniere, indentare, spaţiere, efecte speciale şi proprietăţi de caractere.

În aplicaţia **Word**, fiecărui stil i se atribuie un nume. Denumirea stilului aplicat fiecărui paragraf poate fi vizualizată în partea stângă a ecranului în modul de afişare **Draft** din meniul **View** (*fig. 1.39*, p. 54).

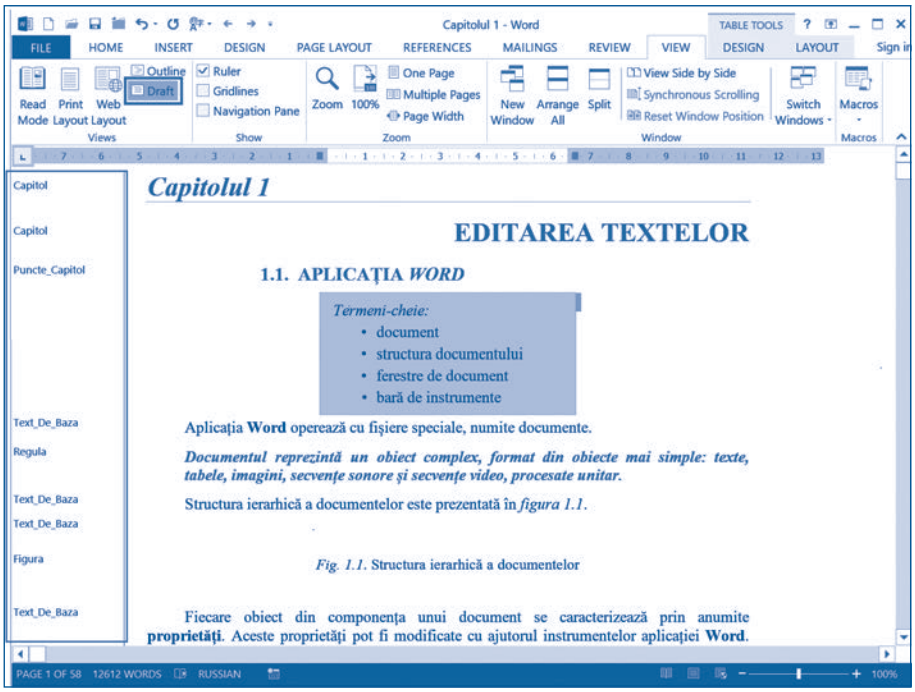


Fig. 1.39. Afişarea stilurilor paragrafelor din text

Din *figura 1.39* se observă că stilurile utilizate la tehnoredactarea acestui manual sunt:

Capitol – stil de paragraf utilizat pentru formatarea titlurilor de capitole.

Puncte_Capitol – stil de paragraf utilizat pentru formatarea titlurilor de compartimente din fiecare capitol.

Text_De_Baza – stil de paragraf cu ajutorul căruia este formatat textul manualului.

Regula – stilul de paragraf utilizat pentru a pune în evidenţă regulile şi definiţiile din manual.

Figura – stilul utilizat pentru formatarea denumirilor de figuri.

Accentuăm faptul că un stil de paragraf include atât proprietăţile paragrafului, cât şi proprietăţile caracterelor din componenţa acestuia. De obicei, stilurile se elaborează de pictori şi de specialişti în domeniul tehnoredactării. În caz de necesitate, utilizatorii avansaţi pot crea şi stiluri proprii. Stilurile disponibile sunt indicate în lista derulantă **Styles** (Stiluri) din panglica de meniuri **Home** (*fig. 1.40*).

Pentru a formata un fragment de text, utilizatorul selectează porţiunea respectivă de document şi indică stilul dorit din listă. Fiecare denumire din această listă este scrisă conform stilului pe care îl reprezintă. Simbolul "¶" indică stilurile de paragraf, iar simbolul "a" – stilurile de caractere. Liniile orizontale ale butoanelor din meniul **Paragraph** simbolizează alinierea paragrafului: la stânga, la dreapta, pe centru, la ambele margini.

În exemplul din *figura 1.40*, documentul conţine stilurile create de utilizator **Adresare**, **Câmp**, **Către**, **Data & Semnătura** şi stilurile **Heading 1** şi **Normal**, încorporate în aplicaţia **Word**. Se observă că **Adresare** este stil de paragraf, iar **Câmp** – stil de caracter.

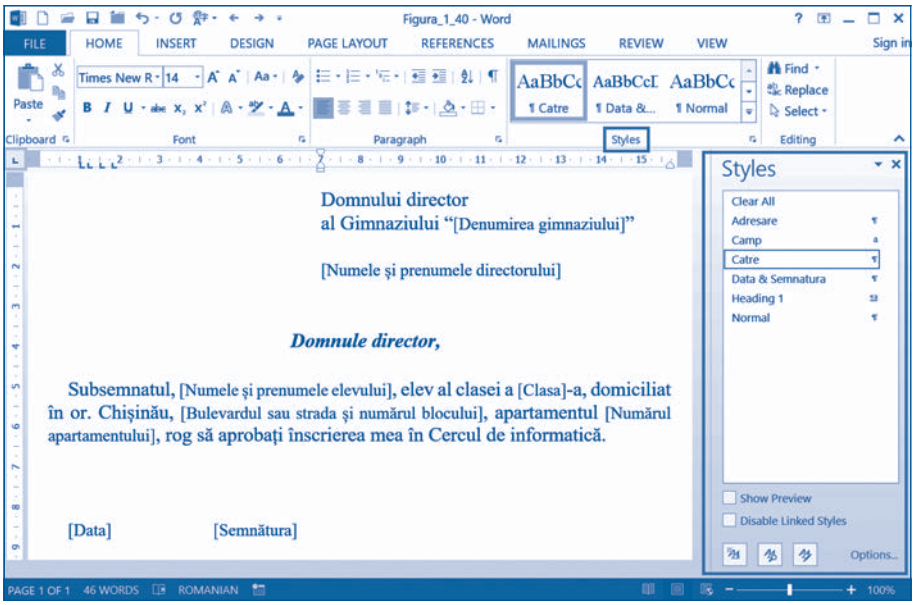


Fig. 1.40. Meniul **Styles**

Proprietăţile stilurilor din documentul curent pot fi vizualizate poziţionând cursorul pe denumirile acestora din fereastra de dialog **Styles**. Modificarea proprietăţilor fiecăruia dintre stiluri se face cu ajutorul comenzilor din meniul contextual, care se afişează pe ecran cu ajutorul unui clic-dreapta pe denumirea stilului dorit (*fig. 1.41*).

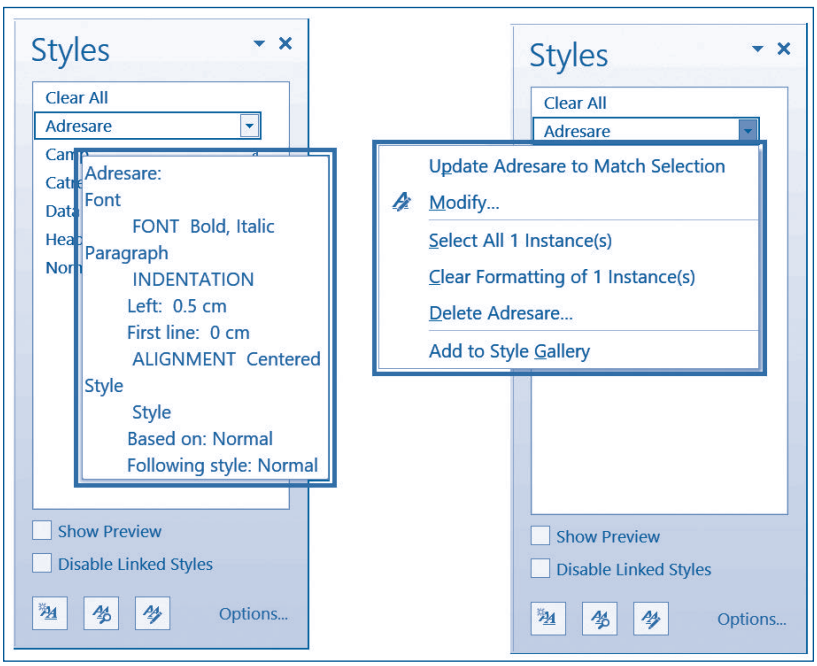


Fig. 1.41. Proprietăţile stilului **Adresare** şi meniul contextual asociat acestuia

Comenzile din meniul contextual permit, de asemenea, ştergerea stilurilor existente, crearea unor stiluri noi şi stabilirea modului de afişare a acestora: în ordine alfabetică, toate sau doar unele stiluri din documentul în curs de elaborare. Mai mult decât atât, în funcţie de configurările sistemului de operare, aplicaţia **Word**, urmărind formatările făcute de utilizator, poate crea în mod automat stiluri noi. Acest fapt duce foarte des la aglomerarea ferestrei de dialog **Styles**. Prin urmare, în scopuri didactice, se recomandă ca utilizatorul să dezactiveze opţiunea de creare automată de noi stiluri, modificând în caz de necesitate setările din meniul **File**, **Options**.

În cazul în care utilizatorul nu indică în mod explicit stilul dorit, aplicaţia va folosi stilul Normal. Proprietăţile de caractere şi de paragraf definite în acest stil depind atât de configurarea sistemului de operare, cât şi de configurarea aplicaţiei Word.

În momentul creării unui document nou, aplicaţia **Word** trebuie să cunoască formatările de pagină, formatările de paragraf şi formatările de caractere. În principiu, utilizatorul poate introduce aceste informaţii pentru fiecare document aparte. Practica ne demonstrează însă că foarte multe documente au formatări similare şi conţin fragmente identice de text. Pentru exemplificare, amintim cererile, Curriculum vitae, foile cu situaţia şcolară, facturile etc. În astfel de cazuri este raţională utilizarea şabloanelor.

***Şabloanele* reprezintă modele ce conţin informaţii despre formatarea paginilor, stiluri de paragraf şi stiluri de caractere, texte şi imagini.**

Fiecare şablon are un nume şi se păstrează într-un fişier cu extensia **.dotx**. De obicei, toate şabloanele disponibile sunt grupate în dosarul **Custom Office Templates** (Şabloanele personalizate a sute de aplicaţii de birou Office).

La lansarea în execuţie sau la acţionarea butonului **New** (Nou) din bara de instrumente standard, aplicaţia **Word** creează un document nou, bazat pe şablonul **Normal** (Obişnuit). Acest şablon conţine formatările de pagină frecvent utilizate şi un stil de paragraf cu aceeaşi denumire. La activarea comenzii **File**, **New** (Fişier, Nou), aplicaţia afişează fereastra de dialog **New**. La selectarea opţiunii **Personal**, pe ecran vor fi afişate pictogramele tuturor şabloanelor din dosarul **Custom Office Templates** (*fig. 1.42*).

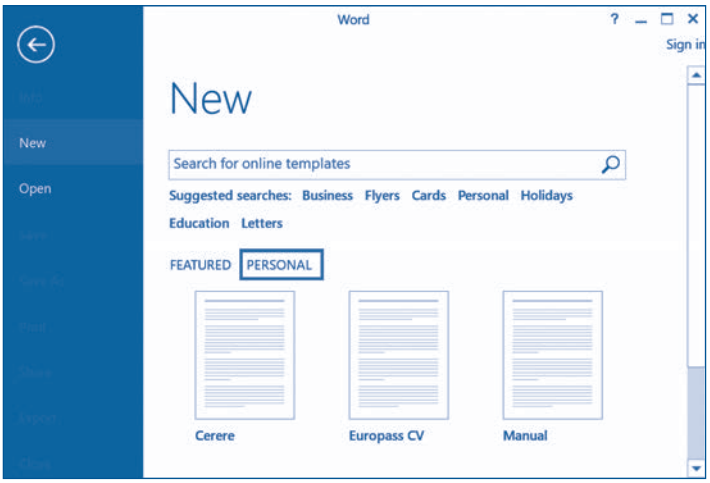


Fig. 1.42. Fereastra de dialog **New**

Utilizatorul poate alege din această fereastră şablonul dorit şi poate crea pe baza lui un document sau un alt şablon. În procesul tehnoredactării aplicaţia **Word** nu face nicio diferenţiere între şabloane şi documente. Tipul fişierului în care va fi salvat conţinutul supus tehnoredactării – document (**.docx**) sau şablon (**.dotx**) – se stabileşte în momentul salvării acestuia. Şablonul se creează numai atunci când utilizatorul indică în fereastra de dialog **Save As** (Salvează ca) opţiunea **Word Template** (Şablon Word).

La selectarea opţiunii **Featured** (Recomandate), în fereastra de dialog **New** vor fi afişate şabloanele recomandate de designeri (*fig. 1.43*). Lista acestora este foarte mare, utilizatorul având astfel posibilitatea să creeze documente cu cele mai diverse aspecte. Mai mult decât atât, caseta de căutare a ferestrei de dialog **New** permite căutarea şabloanelor dorite şi pe Internet, numărul acestora fiind de câteva mii.

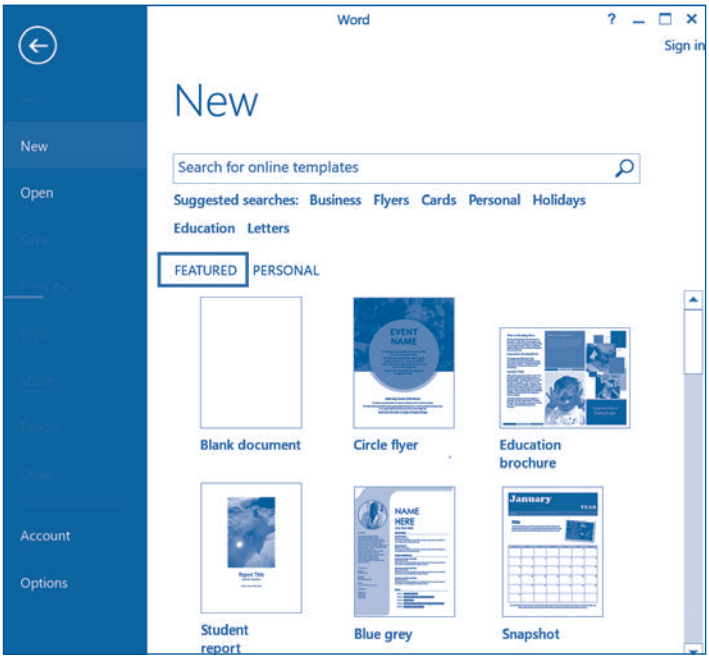


Fig. 1.43. Şabloane recomandate de designeri

Şabloanele standard şi şabloanele elaborate de utilizatori pot fi folosite şi ca **biblioteci de stiluri**. În acest scop, se acţionează butonul **Manage Styles** (Gestionarea stilurilor) din fereastra de dialog **Styles** şi se lansează comanda **Import/Export**. În fereastra respectivă de dialog se indică fişierele din care se vor exporta şi în care se vor importa stilurile dorite.

Prin urmare, utilizatorul poate tehnoredacta documentul curent folosind stiluri din mai multe şabloane.

Întrebări şi exerciţii

❶ Numiţi facilităţile oferite de aplicaţia **Word** pentru crearea şi formatarea documentelor.

❷ **EXPERIMENTEAZĂ!** Afişaţi pe ecran formatările de paragraf şi formatările de caractere din documentele elaborate de dvs.

❸ Explicați destinația pensulei de formatare.

❹ **EXPLOREAZĂ!** Afișați pe ecran denumirile de stiluri utilizate în documentele elaborate de dvs.

❺ Explicați termenii *stil de paragraf* și *stil de caractere*.

❻ Care este destinația șabloanelor? Ce informație conține un șablon?

❼ Afișați pe ecran lista de stiluri ale șablonului **Normal**.

❽ **EXPLOREAZĂ!** Pentru fiecare șablon propus de profesor determinați:
– lista de stiluri disponibile;
– proprietățile stilurilor de paragraf;
– proprietățile stilurilor de caractere.

❾ **CREEAZĂ!** Elaborați un document în baza șablonului **Normal**. Adăugați la lista de stiluri ale acestui document stilurile din șabloanele indicate de profesor.

❿ **ÎNVAȚĂ SĂ ÎNVEȚI!** Utilizând opțiunea **Education** a ferestrei de dialog **New**, afișați pe ecran șabloanele din domeniul educației. Aflați destinația fiecărui șablon. Modificați șabloanele respective, adaptându-le la specificul țării noastre și localității în care se află școala în care învățați. Utilizați șabloanele modificate pentru crearea și diseminarea documentelor folosite în viața dvs. școlară: rapoarte, scrisori, anunțuri, foi volante, pliante etc.

⓫ **CREEAZĂ!** Elaborați șablonul ce urmează. Afișați pe ecran stilurile de paragraf și stilurile de caractere utilizate în acest șablon.

Domnule director,

Subsemnatul, [Numele și prenumele elevului], elev al clasei a [Clasa]-a, domiciliat în or. Chișinău, [Bulevardul sau strada și numărul blocului], apartamentul [Numărul apartamentului], rog să aprobați înscrierea mea la Cercul de Informatică.

Domnului director
al Gimnaziului "[Denumirea gimnaziului]"

[Numele și prenumele directorului]

[Data] [Semnătura]

⓬ **EXPLOREAZĂ!** Afișați pe ecran documentul din *figura 1.40* în regimul de vizualizare **Draft**. Poziționând cursorul pe diferite obiecte din text (caractere, cuvinte, rânduri, paragrafe), aflați destinația elementelor de control din fereastra de dialog **Styles**.

1.13. Corespondența combinată

Termeni-cheie:
- document principal
- sursă de date
- corespondență combinată

Practica ne-a demonstrat că foarte des multe documente au un conținut aproape identic, diferența dintre ele reducându-se la două-trei propoziții. De exemplu, presupunem că este constituit un comitet pentru organizarea Concursului raional la Informatică. Comitetul a elaborat formularul (modelul) unei scrisori-invitații:

«Instituția»
«Adresa»

Dragi prieteni,

Vă invităm să participați la Concursul raional de Informatică.

Organizatorii pot asigura cazarea și masa a *«Componența_echipei»* persoane.

Comitetul organizatoric

Această scrisoare trebuie transmisă la circa 40 de instituții şcolare din raion. Datele despre fiecare instituție şcolară au fost trecute într-un tabel de forma:

Instituția	Adresa	Componența echipei
Gimnaziul "Mihai Moraru"	satul Brăneşti, raionul Orhei	4
Liceul "Ion Luca Caragiale"	strada Vasile Lupu, nr. 60/1, or. Orhei	38
Gimnaziul Chiperceni	satul Chiperceni, raionul Orhei	3
Liceul "Alexandru Donici"	str. Mihai Eminescu, nr. 25, satul Peresecina	6

O soluție aparent simplă a acestei probleme constă în crearea unei scrisori în care se indică denumirea, adresa şi numărul de participanți din Gimnaziul "Mihai Moraru". După tipărirea primei scrisori, ea poate fi modificată, indicându-se informațiile despre Liceul "Ion Luca Caragiale". După tipărirea scrisorii a doua se introduc datele referitoare la Gimnaziul Chiperceni ş.a.m.d. Evident, editarea separată a circa 40 de scrisori, câte una pentru fiecare instituție şcolară, este un proces obositor şi ineficient.

O soluție mult mai bună constă în crearea automată a setului necesar de scrisori. În acest scop, se folosesc comenzile grupului de meniuri **Mailings** (Corespondența). Aceste comenzi utilizează următoarele obiecte:

Main Document (Documentul principal) – reprezintă formularul (modelul) scrisorii. În acest formular informația ce se schimbă de la o scrisoare la alta este codificată cu ajutorul unor simboluri speciale, numite *câmpuri*. De exemplu, formularul examinat mai sus conține câmpurile *«Instituția»*, *«Adresa»* şi *«Componența_echipei»*.

Data Source (Sursa de date) – reprezintă un tabel în care primul rând conține denumirile de câmpuri. Celelalte rânduri ale tabelului conțin datele ce vor fi inserate în locul câmpurilor din documentul principal.

Letters (Corespondența combinată) – un fişier **Word** ce conține câte o scrisoare pentru fiecare rând al tabelului din sursa de date.

Documentul principal poate fi introdus în calculator, editat şi salvat exact la fel ca şi celelalte documente ale aplicației **Word**. Însă, înainte de a insera în el câmpurile dorite, documentul principal trebuie conectat la sursa de date.

De obicei, în calitate de sursă de date se folosesc tabelele create în aplicația **Access**. Pentru exemplificare, în *figura 1.44* (p. 60) este prezentată sursa de date ce conține lista instituțiilor, numărul elevilor din fiecare instituție care vor fi invitați la Concursul raional de Informatică.

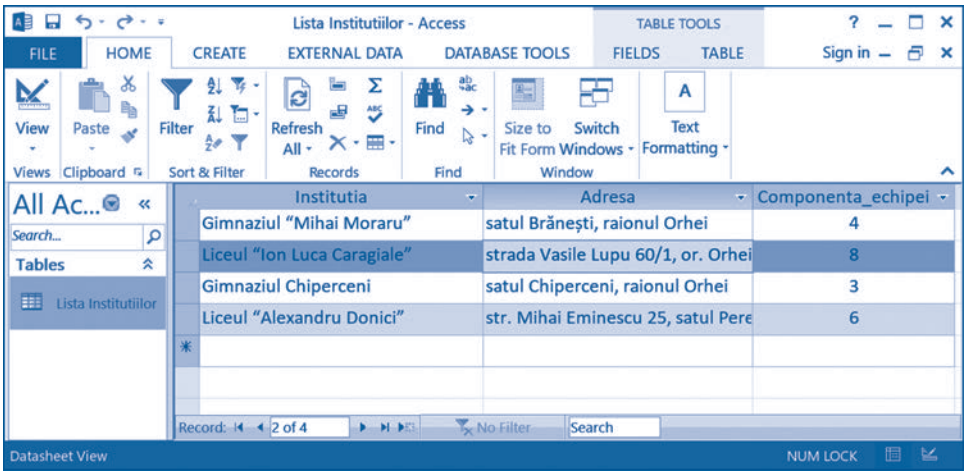


Fig. 1.44. Sursa de date **Lista Instituţiilor**

Inserarea câmpurilor se efectuează cu ajutorul comenzii **Insert Merge Field** (Inserarea câmpului de combinare). După poziționarea cursorului în locul în care se dorește inserarea unui câmp şi lansarea comenzii respective, din fereastra de dialog afişată pe ecran (*fig. 1.45*) se alege câmpul dorit.

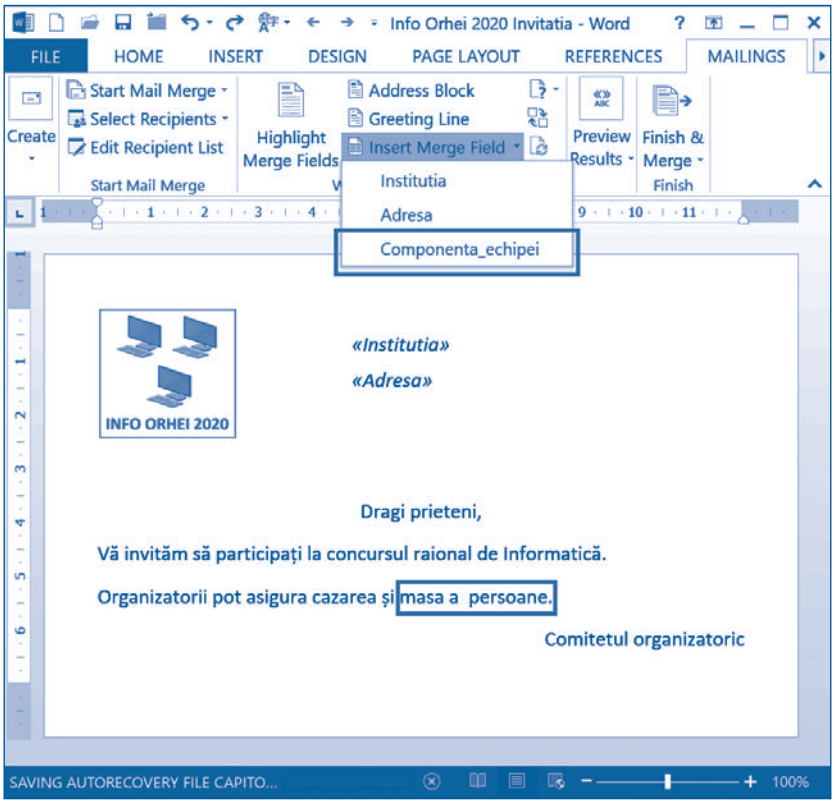


Fig. 1.45. Inserarea câmpurilor de combinare

Menționăm că denumirile de câmpuri trebuie să fie formate dintr-un singur cuvânt. În caz de necesitate, câteva cuvinte pot fi reunite în unul singur cu ajutorul caracterului "_", de exemplu, «*Componența_echipei*».

Pentru a crea corespondența combinată, se parcurg consecutiv următorii paşi:

1) se creează fişierul care conține documentul principal, de exemplu scrisoarea-invitație la Concursul raional de Informatică;

2) se creează fişierul ce conține sursa de date, de exemplu tabelul cu datele despre instituțiile şcolare;

3) documentul principal şi sursa de date se îmbină cu ajutorul comenzilor din meniul **Finish** & **Merge**.

În urma îmbinării se va obține câte o scrisoare separată pentru fiecare rând al tabelului din sursa de date. Aceste scrisori vor fi inserate într-un document nou (*fig. 1.46*), care poate fi salvat sau tipărit la imprimantă.

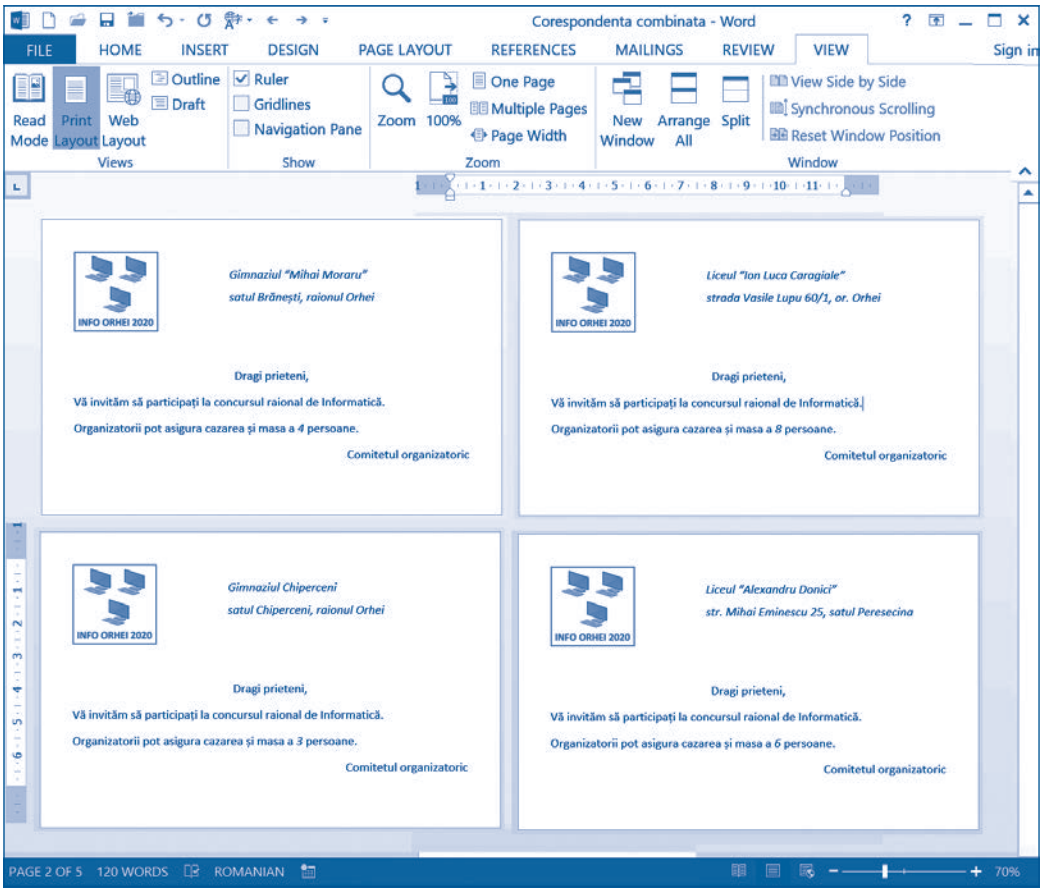


Fig. 1.46. Corespondență combinată

În general, grupul de meniuri **Mailings** permite cumularea tuturor operațiilor destinate creării corespondenței combinate. Principalele facilități oferite de aceste meniuri sunt:

– crearea şi editarea documentelor principale pentru scrisori, plicuri, etichete poştale şi scrisori electronice;

– crearea şi editarea surselor de date cu ajutorul unor formulare speciale;

– utilizarea surselor de date create cu ajutorul altor aplicaţii ale sistemului de operare;

– crearea corespondenţei combinate numai pentru anumite rânduri ale tabelului din sursa de date.

Evident, aceste instrumente pot fi folosite nu numai pentru scrisori, dar şi pentru alte documente ce au un conţinut aproape identic: situaţia şcolară a fiecărui elev, note de plată, foi de trăsură etc.

Întrebări şi exerciţii

❶ Daţi exemple de documente care pot fi elaborate cu ajutorul instrumentelor de corespondenţă combinată.

❷ Explicaţi termenul *document principal*. Prin ce se deosebeşte documentul principal de alte documente ale aplicaţiei **Word**?

❸ Ce informaţii conţine sursa de date? Cum este organizată această informaţie?

❹ Explicaţi termenul *câmp*. Care este destinaţia câmpurilor dint-un document?

❺ Ce informaţie conţine un document de tipul *corespondenţă combinată*? Ce operaţii efectuează calculatorul pentru a crea un astfel de document?

❻ **EXPLOREAZĂ!** Afişaţi pe ecran grupul de meniuri **Mailings**. Aflaţi cu ajutorul sistemului de asistenţă destinaţia fiecărui element de control din aceste meniuri.

❼ **CREEAZĂ!** Reprezentaţi pe un desen fluxurile de date care circulă între obiectele corespondenţei combinate: documentul principal, sursa de date şi corespondenţa combinată.

❽ **ÎNVAŢĂ SĂ ÎNVEŢI!** Utilizând sistemul de asistenţă, aflaţi modul de creare a documentelor pentru tipărirea plicurilor şi etichetelor poştale, pentru transmiterea automată a scrisorilor electronice.

❾ **EXPERIMENTEAZĂ!** Creaţi documentul principal **Scrisoare-invitaţie** şi sursa de date **Instituţii şcolare** studiate în acest paragraf. Memorizaţi într-un fişier şi afişaţi pe ecran corespondenţa combinată din *figura 1.46*.

❿ **LUCREAZĂ ÎN ECHIPĂ!** Creaţi conform modelului ce urmează documentul **Situaţia şcolară** pentru fiecare elev din clasă:

Situaţia şcolară

a elevului (elevei).....

1. Limba română .....
2. Limba modernă .....
3. Matematica .....
4. Fizica .....
5. Istoria românilor .....

...

Diriginte .....

Capitolul 2

ALGORITMI

2.1. Algoritmi și executanți

Termeni-cheie:
- algoritm
- executant
- comandă manuală
- comandă prin program
- program
- limbaj de programare

Prelucrarea informației cu ajutorul calculatoarelor presupune existența unor programe care se încarcă în memoria internă a acestora. Cunoaştem deja că orice program de calculator reprezintă o succesiune de cuvinte binare care indică ordinea (secvența) calculelor. Programele pentru calculatoare se elaborează în baza algoritmilor.

În mod intuitiv, *algoritmul* reprezintă o mulțime finită de instrucțiuni care, fiind executate într-o ordine bine stabilită, produc în timp finit un rezultat. Procesul de elaborare a algoritmilor se numeşte *algoritmizare*.

Amintim că în viața cotidiană instrucțiunile reprezintă anumite indicații date cuiva în vederea îndeplinirii unui lucru. Exemple de instrucțiuni:

1) înainte de a porni calculatorul, verificați integritatea cablului de rețea;

2) dacă $a \neq 0$, calculați $x = b/a$;

3) pentru a afla rădăcinile ecuației de gradul doi, calculați mai întâi discriminantul $D = b^2 - 4ac$.

Cuvântul **algoritm** provine de la numele marelui matematician al Evului Mediu *Al-Khwarizmi Muhammed ibn Musa* (cca 780–850), care a folosit pentru prima oară reguli precise şi clare pentru efectuarea operațiilor aritmetice de bază, studiate astăzi în clasele primare: adunarea, scăderea, înmulțirea şi împărțirea. Mai târziu, aceste reguli apar sub denumirea de algoritm în cartea "Elementele lui Euclid". Se consideră că algoritmul lui Euclid pentru calculul celui mai mare divizor comun a două numere naturale este unul dintre primii algoritmi cunoscuți în matematică.

Exemplu de algoritm:

Date inițiale: lungimile a, b ale catetelor triunghiului dreptunghic.
1. Calculați $x=a^2$.
2. Calculați $y=b^2$.

3. Calculați $z=x+y$.
4. Calculați $c=\sqrt{z}$.
Rezultat: lungimea ipotenuzei c.

Evident, acest algoritm poate fi executat de orice persoană care cunoaşte operațiile aritmetice de bază, operațiile cu puteri şi operațiile cu radicali.

În prezent, sensul cuvântului *algoritm* s-a extins, acest termen fiind utilizat în mai multe domenii ale ştiinței şi tehnicii pentru a descrie o succesiune de instrucțiuni destinate soluționării anumitor probleme. În informatică noțiunea de algoritm se asociază în mod obligatoriu cu noțiunea de executant.

***Executantul* reprezintă un obiect care poate îndeplini anumite comenzi. Mulțimea acestor comenzi formează repertoriul executantului.**

De exemplu, televizorul poate fi tratat ca un obiect care îndeplineşte următoarele comenzi: conectarea la rețea, deconectarea de la rețea, mărirea volumului, micşorarea volumului, recepționarea canalului nr. 1, recepționarea canalului nr. 2 ş.a.m.d. Comenzile respective sunt comunicate televizorului prin intermediul butoanelor de pe pupitrul de comandă. În mod similar, calculatorul reprezintă un obiect care îndeplineşte următoarele comenzi: adunarea, scăderea, înmulțirea, împărțirea şi compararea numerelor, citirea datelor de la tastatură, afişarea datelor pe ecran, scrierea datelor pe un disc magnetic, citirea datelor de pe un disc optic etc.

Definirea exactă a unui executant include:

1) descrierea setului (repertoriului) de comenzi pe care executantul le poate îndeplini;

2) descrierea mediului în care lucrează executantul.

În continuare vom studia executanții **Cangurul** şi **Furnica**, elaborați în scopuri didactice de un grup de elevi şi profesori de Informatică din țara noastră.

Executantul Cangurul

Acest executant reprezintă un program de calculator, care derulează sub sistemul de operare **Windows**. Fereastra aplicației **Cangurul** este prezentată în *figura 2.1*.

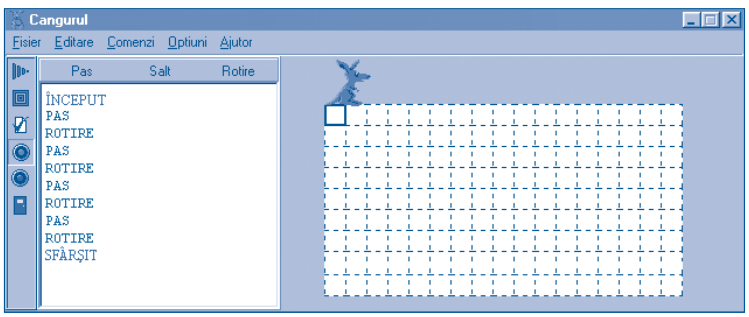


Fig. 2.1. Fereastra aplicației **Cangurul**

Executantul propriu-zis este simbolizat prin pictograma unui cangur, care poate îndeplini următoarele comenzi:

PAS – Cangurul se deplasează cu un pătrățel, trasând segmentul respectiv de dreaptă;

SALT – Cangurul se deplasează cu un pătrățel, însă nu desenează nimic;
ROTIRE – Cangurul se roteşte cu 90° după ácele de ceasornic.

Fereastra aplicației **Cangurul** conține următoarele elemente:

1) **Bara de meniuri**, care include meniurile standard Fişier, Editare, Comenzi, Opțiuni, Ajutor;

2) **Centrul de comandă**, care include butoanele Pas, Salt, Rotire, Execută, Stop, Control, Executare manuală, Executare automată, Ieşire;

3) **Zona de editare** a programelor;

4) **Mediul de lucru** al Cangurului, care reprezintă un câmp dreptunghiular, liniat în pătrățele.

Deosebim două moduri de comandă a executanților: comandă manuală şi comandă prin program.

***Modul de comandă manuală* presupune introducerea separată a fiecărei comenzi şi îndeplinirea ei de către executant.**

De exemplu, presupunem că se doreşte desenarea unui pătrat. În cazul comenzii manuale, utilizatorul va acționa butoanele Pas, Rotire şi Salt din componența Centrului de comandă în aşa mod, încât Cangurul să deseneze pătratul respectiv.

***Modul de comandă prin program* presupune memorarea în prealabil a unei secvențe de comenzi şi executarea lor în regim automat, fără intervenția utilizatorului.**

În cazul modului de comandă prin program, utilizatorul va memora în zona de editare a programelor secvența de instrucțiuni (comenzi) care asigură desenarea pătratului (*fig. 2.1*). Să reținem că începutul şi sfârşitul fiecărui program se indică cu ajutorul cuvintelor auxiliare ÎNCEPUT şi SFÂRŞIT.

***Programul* reprezintă un algoritm scris în limbajul executantului. Procesul de elaborare a programelor se numeşte *programare*.**

Evident, programele pot fi salvate pe disc şi utilizate de mai multe ori. În cazul aplicației **Cangurul**, salvarea şi deschiderea programelor se efectuează cu ajutorul comenzilor grupate în meniul Fişier.

În general, programele pentru executanți se scriu cu ajutorul unor limbaje speciale, denumite **limbaje de programare**. Aceste limbaje conțin instrucțiuni şi cuvinte auxiliare. De obicei, comenzile executantului sunt în acelaşi timp şi instrucțiuni ale limbajului de programare. De exemplu, limbajul de programare al executantului **Cangurul** conține instrucțiunile PAS, SALT, ROTIRE şi cuvintele auxiliare ÎNCEPUT şi SFÂRŞIT.

Executantul Furnica

În această aplicație executantul este simbolizat prin imaginea unei furnici, care se poate deplasa pe un câmp divizat în pătrățele (*fig. 2.2*, p. 66). În unele pătrățele ale câmpului pot fi inserate caractere imprimabile. De obicei, se cere elaborarea unui program care aranjează aceste caractere într-o anumită ordine.

Executantul poate îndeplini comenzile SUS, JOS, DREAPTA, STÂNGA, care deplasează Furnica din pătrățelul curent în unul dintre pătrățelele vecine. Dacă în pătrățelul vecin se află un caracter, el va fi împins, când acest lucru este posibil, în direcția

mişcării. Începutul şi sfârşitul programelor se indică cu ajutorul cuvintelor auxiliare ÎNCEPUT şi SFÂRŞIT.

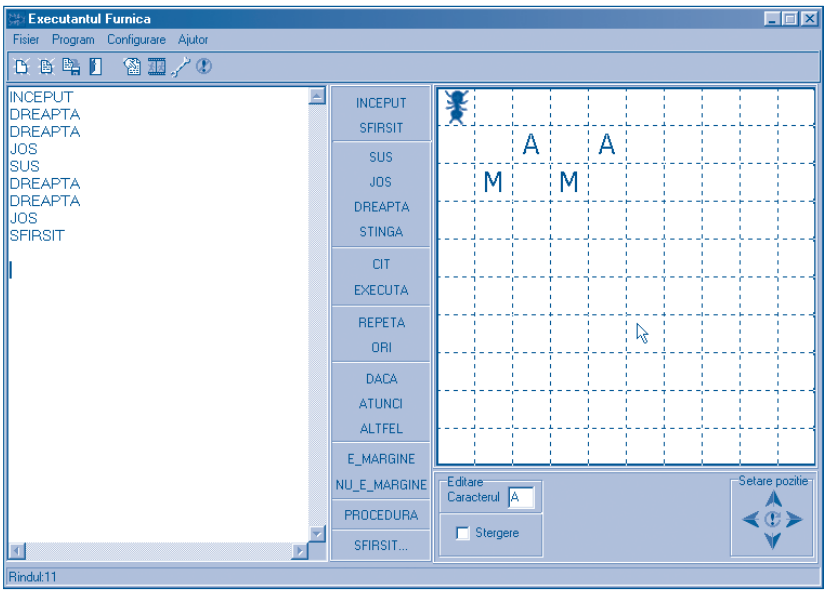


Fig. 2.2. Fereastra aplicației **Furnica**

Fereastra aplicației **Furnica** (*fig. 2.2*) conține următoarele elemente:

1) **Bara de meniuri**, care include meniurile standard Fişier, Program, Configurare, Ajutor;

2) **Butoanele** Nou, Deschide, Salvare, Ieşire, Analiza sintactică, Execută, Configurare, Stop;

3) **Zona de editare** a programelor;

4) **Mediul de lucru** al Furnicii, care reprezintă un câmp dreptunghiular, liniat în pătrățele;

5) **Centrul de comandă**, care include butoanele SUS, JOS, DREAPTA, STÂNGA, simbolizate prin săgeți;

6) **Caseta de inserare** a caracterelor imprimabile;

7) **Butoanele** ÎNCEPUT, SFÂRŞIT, SUS, JOS, ..., PROCEDURA, destinate simplificării proceselor de editare a programelor.

În modul de comandă manuală, utilizatorul acționează săgețile care simbolizează comenzile SUS, JOS, DREAPTA, STÂNGA. În modul de comandă prin program, utilizatorul introduce mai întâi instrucțiunile respective în zona de editare. După definitivare, programul poate fi lansat în execuție şi/sau salvat pe disc. Pentru exemplificare, în *figura 2.2* este prezentat programul care aranjează literele din celulele mediului de lucru al Furnicii în aşa mod, încât ele să formeze cuvântul MAMA.

Accentuăm faptul că, indiferent de tipul executantului, programele servesc ca "date inițiale" pentru Centrul de comandă, care parcurge instrucțiunile programului în curs de derulare şi generează în baza lor comenzi pentru executant. În paragrafele ce urmează vom vedea că limbajele de programare conțin instrucțiuni speciale, care pot genera sute şi mii de comenzi, descriind astfel într-o formă compactă prelucrări foarte complexe.

Întrebări şi exerciţii

❶ Amintiţi-vă cel puţin trei algoritmi pe care i-aţi studiat la lecţiile de matematică.

❷ Pot fi oare reţetele culinare tratate ca algoritmi? Argumentaţi răspunsul dvs.

❸ **CREEAZĂ!** Încercaţi să formulaţi algoritmii destinaţi adunării, scăderii, înmulţirii şi împărţirii numerelor zecimale. Formulaţi astfel de algoritmi pentru adunarea şi scăderea numerelor binare.

❹ Încercaţi să alcătuiţi o listă completă a comenzilor pe care le poate executa televizorul dvs.

❺ Alcătuiţi o listă completă a comenzilor pe care le poate executa un *player* (aparat pentru redarea înregistrărilor).

❻ Ce fel de informaţii trebuie să conţină o descriere completă a unui executant?

❼ Descrieţi repertoriul de comenzi şi mediul de lucru al executantului **Cangurul**.

❽ Descrieţi repertoriul de comenzi şi mediul de lucru al executantului **Furnica**.

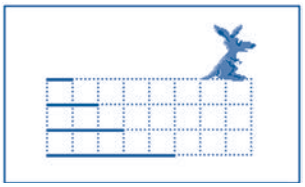
❾ Care este diferenţa dintre algoritmi şi programe? Argumentaţi răspunsul dvs.

❿ Prin ce se deosebeşte modul de comandă prin program de modul de comandă manuală?

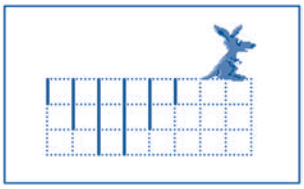
⓫ **EXPLOREAZĂ!** Lansaţi în execuţie aplicaţia **Cangurul**. Aflaţi cu ajutorul sistemului de asistenţă destinaţia tuturor elementelor de control din fereastra acestei aplicaţii.

⓬ **CREEAZĂ!** Lansând comenzile PAS, `SALT`, `ROTIRE`, desenaţi figurile ce urmează:

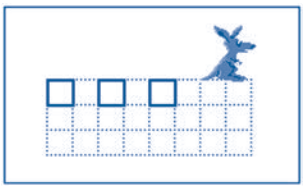
a)



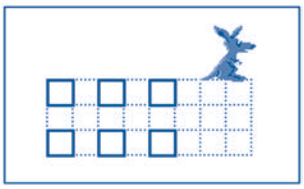
b)



c)

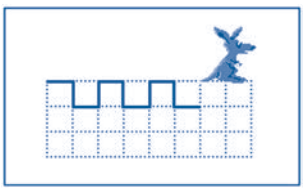


d)

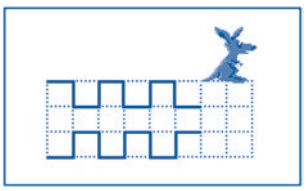


⓭ **ELABOREAZĂ!** Scrieţi programele pentru desenarea figurilor ce urmează:

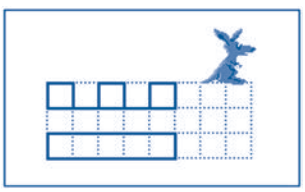
a)



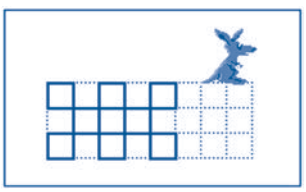
b)



c)



d)

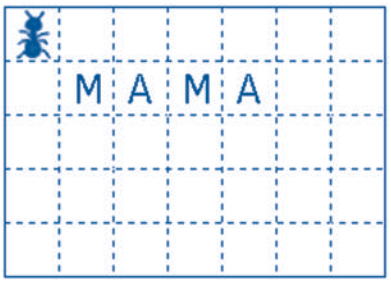


Salvaţi aceste programe pe un suport extern de memorie.

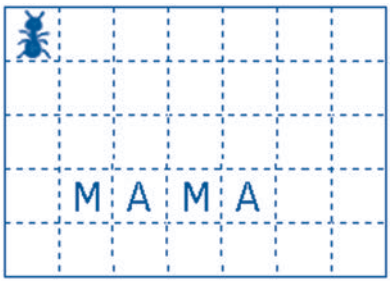
⓮ **EXPLOREAZĂ!** Lansați în execuție aplicația **Furnica**. Aflați cu ajutorul sistemului de asistență destinația tuturor elementelor de control din fereastra acestei aplicații.

⓯ **CREEAZĂ!** Lansând comenzile `SUS`, `JOS`, `DREAPTA`, `STÂNGA`, aranjați caracterele din *figura 2.2* conform modelelor ce urmează:

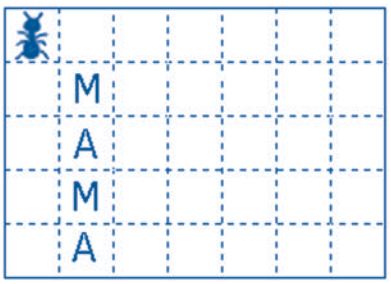
a)



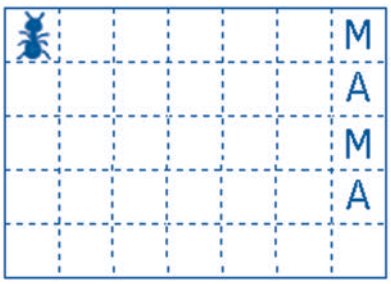
b)



c)



d)



⓰ **ELABOREAZĂ!** Scrieți programele destinate aranjării caracterelor din *figura 2.2* conform modelelor prezentate în exercițiul 15. Salvați aceste programe pe un suport extern de memorie.

⓱ Reprezentați pe un desen figura trasată de executantul **Cangurul** după execuția programului ce urmează:

1	**2**	**3**
ÎNCEPUT	PAS	PAS
PAS	ROTIRE	PAS
PAS	PAS	ROTIRE
ROTIRE	PAS	SFÂRȘIT
PAS	ROTIRE	

⓲ Reprezentați pe un desen amplasarea literelor din *figura 2.2* după execuția programului ce urmează:

1	**2**	**3**
ÎNCEPUT	JOS	STÂNGA
JOS	DREAPTA	STÂNGA
DREAPTA	DREAPTA	STÂNGA
DREAPTA	SUS	SUS
STÂNGA	DREAPTA	DREAPTA
STÂNGA	SUS	DREAPTA
JOS	JOS	SFÂRȘIT

⓳ Care este rolul Centrului de comandă în procesul execuţiei unui program?

⓴ Cine dă comenzi executantului în modul de comandă manuală? Dar în modul de comandă prin program?

2.2. Subalgoritmi

Termeni-cheie:
- subprogram
- programul principal
- procedură
- apel de procedură
- rafinare succesivă

Ca şi în cazul limbajelor folosite de oameni, un limbaj de programare se defineşte prin sintaxa şi semantica lui. E cunoscut faptul că sintaxa este un set de reguli care guvernează alcătuirea propoziţiilor, iar semantica este un set de reguli care determină înţelesul, semnificaţia propoziţiilor respective. Evident, în cazul limbajelor de programare echivalentul propoziţiei este programul.

Sintaxa programelor poate fi descrisă cu ajutorul **formatelor** care reprezintă modele de utilizare a instrucţiunilor, a simbolurilor şi a cuvintelor auxiliare. De exemplu, în cazul executanţilor **Cangurul** şi **Furnica**, programele scrise până în prezent aveau formatul ce urmează:

```
ÎNCEPUT
Instrucţiunea_1
Instrucţiunea_2
...
Instrucţiunea_k
SFÂRŞIT
```

unde *Instrucţiunea_k*, *k* = 1, 2, 3 ş.a.m.d. poate fi oricare din comenzile PAS, SALT, ROTIRE în cazul executantului **Cangurul** şi SUS, JOS, DREAPTA, STÂNGA în cazul executantului **Furnica**.

Analiza programelor de o reală importanţă practică a scos în evidenţă faptul că foarte multe dintre ele conţin secvenţe de instrucţiuni care execută aceleaşi operaţii. De exemplu, presupunem că se doreşte desenarea a opt pătrate, aranjate conform *figurii 2.3*.

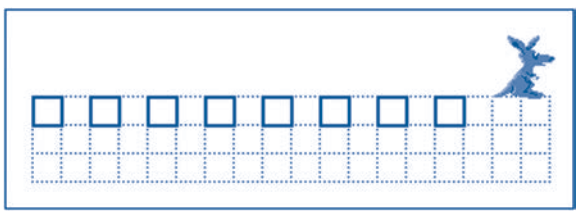


Fig. 2.3. Desenarea pătratelor

Soluționând această problemă prin metoda forței brute, am putea scrie un program în care fiecare pătrat va fi desenat trasându-i-se laturile respective cu ajutorul instrucțiunilor PAS și ROTIRE. Evident, un astfel de program va fi foarte lung, iar scrierea lui va reprezenta o muncă obositoare. O soluție mai elegantă constă în elaborarea unui program auxiliar pentru desenarea unui pătrat de dimensiunile dorite și folosirea ulterioară a acestui program pentru desenarea unui număr arbitrar de pătrate. În astfel de cazuri, programul auxiliar se numește **subprogram**, iar programul care îl apelează – **program principal** sau, pur și simplu, **program**.

În cazul executanților **Cangurul** și **Furnica**, subprogramele se numesc **proceduri** și au următorul format:

```
PROCEDURA Nume
Instrucțiunea_1
Instrucțiunea_2
...
Instrucțiunea_k
SFÂRȘITUL PROCEDURII
```

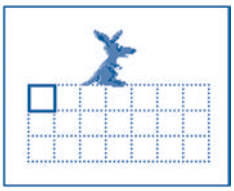
Numele procedurii reprezintă o secvență formată din litere și cifre, care începe, în mod obligatoriu, cu o literă. Cuvântul PROCEDURA nu este o instrucțiune, ci un cuvânt auxiliar, care comunică Centrului de comandă numele procedurii și marchează locul de unde începe descrierea ei. Sfârșitul acestei descrieri este indicat de cuvintele auxiliare SFÂRȘITUL PROCEDURII. Instrucțiunile încadrate între cuvintele auxiliare PROCEDURA și SFÂRȘITUL PROCEDURII formează **corpul procedurii**.

Exemplu:

Procedura Pătrat

```
PROCEDURA Pătrat
PAS
ROTIRE
PAS
ROTIRE
PAS
ROTIRE
PAS
ROTIRE
SALT
SALT
SFÂRȘITUL PROCEDURII
```

Figura desenată



Procedura Pătrat desenează un pătrat, poziția inițială a Cangurului fiind în colțul stânga-sus. Poziția finală a Cangurului a fost aleasă în așa mod, încât ea coincide cu poziția inițială a pătratului următor (*fig. 2.3*).

Instrucțiunile din componența procedurilor vor fi executate numai atunci când în procesul derulării programului principal se va întâlni instrucțiunea **apel de procedură**. Această instrucțiune are următorul format:

```
EXECUTĂ Nume
```

unde cuvântul *Nume* reprezintă numele procedurii apelate.

Când Centrul de comandă întâlneşte un apel de procedură, execuția programului principal se suspendă şi începe îndeplinirea instrucțiunilor din componența procedurii apelate. După terminarea acestor instrucțiuni, se reia executarea programului principal. Prin urmare, programul `Opt_pătrate` pentru desenarea pătratelor din *figura 2.3* va include instrucțiunile ce urmează:

```
ÎNCEPUT
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
EXECUTĂ Pătrat
SFÂRŞIT
```

Evident, programul principal trebuie să cunoască descrierea tuturor procedurilor care vor fi apelate. În funcție de tipul executantului, procedurile respective pot fi incluse în programul principal sau înscrise în fişiere separate pe discurile magnetice.

În cazul executanților **Cangurul** şi **Furnica**, procedurile se includ la începutul programului principal. Pentru aceşti executanți, formatul general al unui program este:

```
PROCEDURA Nume_1
...
SFÂRŞITUL PROCEDURII
PROCEDURA Nume_2
...
SFÂRŞITUL PROCEDURII
...
ÎNCEPUT
Instrucțiunea_1
Instrucțiunea_2
...
Instrucțiunea_k
SFÂRŞIT
```

Din formatul prezentat se observă că un program este format din descrierea subprogramelor *Nume_1*, *Nume_2* ş.a.m.d., după care urmează instrucțiunile programului principal, încadrate între cuvintele auxiliare `ÎNCEPUT` şi `SFÂRŞIT`. Instrucțiunile respective şi cuvintele care le încadrează formează corpul programului. Prin urmare, un program este format din **descrierile de subprograme** şi **corpul programului** propriu-zis.

Amintim faptul că programul reprezintă un algoritm scris în limbajul executantului. Evident, clasificarea programelor în subprograme şi programe principale se aplică şi în cazul algoritmilor. În procesul soluționării unei probleme, utilizatorul elaborează algoritmii şi subalgoritmii necesari, scriind programele şi subprogramele respective pentru fiecare tip concret de executant.

În general, subprogramele reprezintă subalgoritmi, destinați soluționării anumitor probleme întâlnite frecvent în procesul elaborării algoritmului principal. Metoda de soluționare a problemelor complexe prin divizarea lor în probleme mai simple

se numeşte **metoda rafinării succesive**. De exemplu, în cazul *figurii 2.3*, problema iniţială – desenarea celor opt pătrate – a fost divizată în opt subprobleme identice – desenarea unui singur pătrat.

Întrebări şi exerciţii

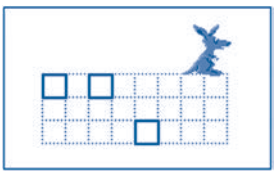
❶ Când apare necesitatea utilizării subalgoritmilor? Daţi exemple.

❷ **STUDIU DE CAZ!** Care este diferenţa dintre un algoritm şi un subalgoritm?

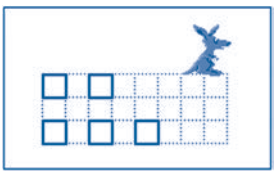
❸ Care este formatul procedurilor scrise în limbajul executanţilor **Cangurul** şi **Furnica**? Cum se apelează aceste proceduri?

❹ **ELABOREAZĂ!** Utilizând procedura `Pătrat`, elaboraţi programele necesare pentru desenarea figurilor ce urmează:

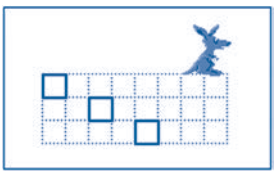
a)



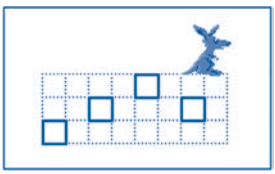
b)



c)

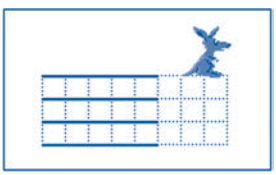


d)

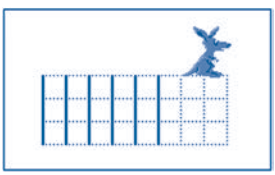


❺ **ANALIZEAZĂ!** Identificaţi în figurile ce urmează fragmentele care se repetă. Scrieţi procedurile necesare pentru desenarea acestor fragmente.

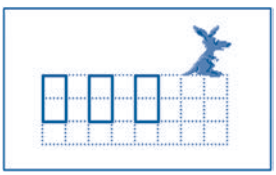
a)



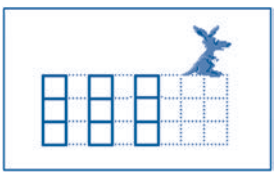
b)



c)



d)



Utilizând procedurile elaborate, scrieţi programele necesare pentru desenarea figurilor prezentate mai sus.

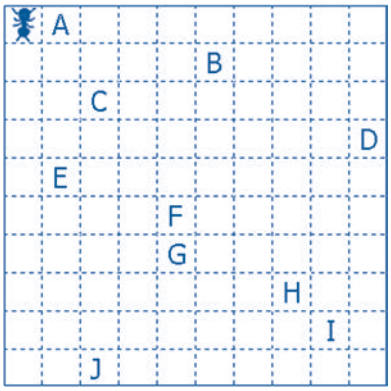
❻ Aplicând metoda rafinării succesive, scrieţi programele pentru desenarea figurilor c şi d din exerciţiul 12 al paragrafului 2.1.

❼ Scrieţi în limbajul executantului **Furnica** următoarele proceduri:
`Stânga_5` – deplasarea Furnicii în STÂNGA cu cinci poziţii;
`Dreapta_5` – deplasarea Furnicii în dreapta cu cinci poziţii;
`Sus_5` – deplasarea Furnicii în sus cu cinci poziţii;
`Jos_5` – deplasarea Furnicii în jos cu cinci poziţii.

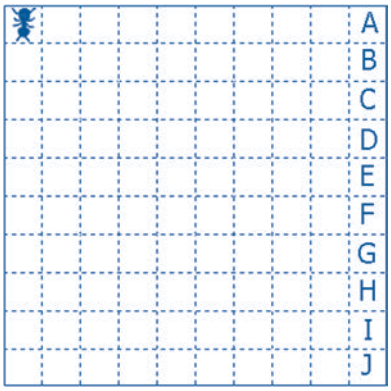
Utilizând aceste proceduri, elaborați un program care deplasează Furnica în așa mod, încât traiectoria mișcării să reprezinte un pătrat cu laturile: *a)* 5 × 5; *b)* 6 × 6; *c)* 8 × 8.

❽ În figurile ce urmează este prezentat aspectul inițial și aspectul final al câmpului de lucru al executantului **Furnica**. Scrieți o procedură care mută caracterul imprimabil dintr-un anumit rând în ultima celulă a rândului.

Aspectul inițial



Aspectul final



Scrieți un program care mută toate caracterele imprimabile în ultima coloană a câmpului.

2.3. Algoritmi repetitivi. Ciclu cu contor

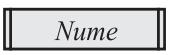
Termeni-cheie:
- schemă logică
- algoritm liniar
- algoritm repetitiv
- ciclu cu contor
- instrucțiune simplă
- instrucțiune compusă

Pentru o reprezentare mai sugestivă, algoritmii pot fi descriși cu ajutorul schemelor logice. **Schema logică** reprezintă un desen, care conține următoarele simboluri grafice:

– punctul de pornire a procesului de execuție a algoritmului;

– punctul de oprire a procesului de execuție a algoritmului;

– execuția unei instrucțiuni;



– apelul unui subalgoritm.

Liniile orientate din componența schemelor logice indică ordinea în care trebuie executate instrucțiunile algoritmului.

Pentru exemplificare, în *figura 2.4* (p. 74) este prezentată schema logică a procedurii `Pătrat` și schema logică a programului `Opt_pătrate`.

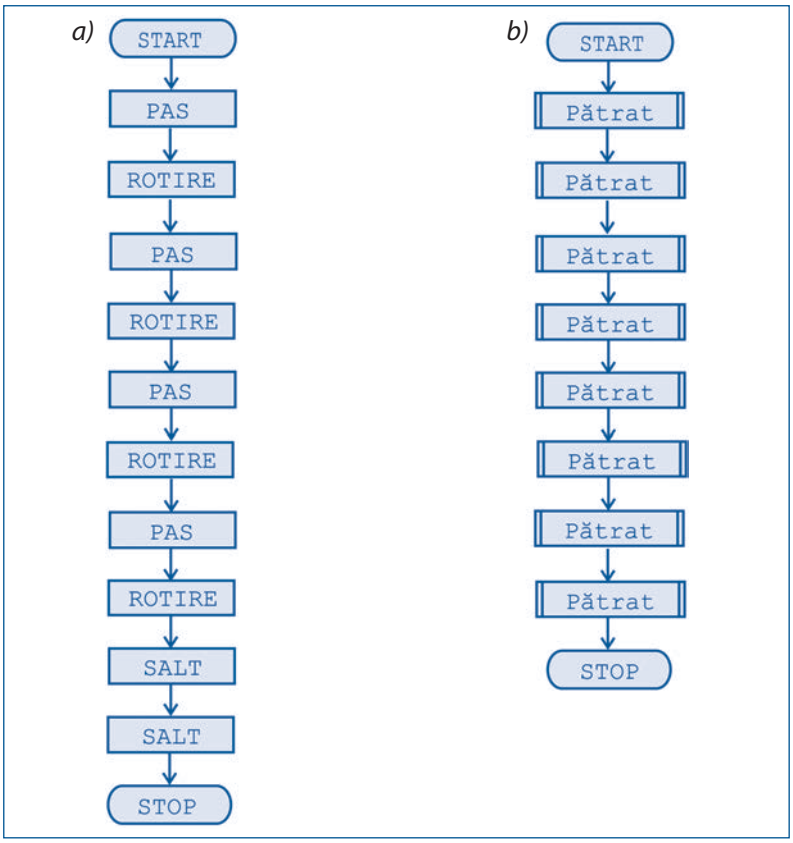


Fig. 2.4. Scheme logice:
a – procedura `Pătrat`; *b* – programul `Opt_pătrate`

Din analiza schemelor logice din *figura 2.4* se observă că procesul de execuție a unui algoritm poate fi simbolizat printr-o deplasare imaginară dintr-un simbol grafic în altul în direcția indicată de liniile respective.

Algoritmii instrucțiunile cărora sunt executate în ordinea apariției lor în text se numesc *algoritmi liniari*.

Evident, în cazul algoritmilor liniari drumul imaginar parcurs de la simbolul grafic `START` până la simbolul grafic `STOP` reprezintă o linie ce nu se autointersectează (*fig. 2.4*).

În procesul elaborării algoritmilor s-a observat că unele secvențe de instrucțiuni deseori trebuie executate de mai multe ori. De exemplu, în cazul procedurii `Pătrat` (*fig. 2.4, a*), secvența de instrucțiuni PAS, `ROTIRE` se execută de patru ori, iar instrucțiunea apel de procedură din programul `Opt_pătrate` – de opt ori. Pentru a simplifica procesele de elaborare a algoritmilor, în astfel de cazuri se poate utiliza instrucțiunea `REPETĂ`. Formatul şi schema logică a acestei instrucțiuni sunt date în *figura 2.5*.

În descrierea instrucțiunii `REPETĂ` *n* reprezintă numărul dorit de repetări, iar cuvintele `REPETĂ`, ORI, `SFÂRŞITUL REPETĂRII` sunt cuvinte auxiliare.

Instrucțiunea `REPETĂ` se notează pe câteva linii şi include în componența sa alte

instrucțiuni. Instrucțiunile de felul acesta se numesc **instrucțiuni compuse**, spre deosebire de **instrucțiunile simple** PAS, SALT, ROTIRE, SUS, JOS, DREAPTA, STÂNGA, apel de procedură, studiate în paragrafele precedente.

```
REPETĂ n ORI
Instrucțiunea_1
Instrucțiunea_2
...
Instrucțiunea_k
SFÂRȘITUL REPETĂRII
```

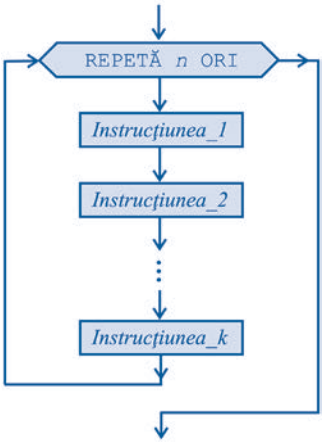


Fig. 2.5. Formatul şi schema logică a instrucțiunii REPETĂ

În procesul execuției instrucțiunii REPETĂ, Centrul de comandă va îndeplini de *n* ori secvența de instrucțiuni încadrată între cuvintele auxiliare. Utilizând această instrucțiune, putem transcrie programele ce conțin secvențe de instrucțiuni, care trebuie executate de mai multe ori, într-o formă mai compactă. De exemplu, programul Opt_pătrate poate fi transcris în forma:

1
```
PROCEDURA Pătrat
REPETĂ 4 ORI
PAS
ROTIRE
SFÂRȘITUL REPETĂRII
SALT
SALT
SFÂRȘITUL PROCEDURII
```

2
```
ÎNCEPUT
REPETĂ 8 ORI
EXECUTĂ Pătrat
SFÂRȘITUL REPETĂRII
SFÂRȘIT
```

Instrucțiunea REPETĂ *n* ORI se numeşte **ciclu cu contor**, deoarece la execuția ei se repetă ciclic aceeaşi secvență de instrucțiuni, iar numărul de repetări *n* este cunoscut în momentul scrierii programului. Secvența de instrucțiuni încadrată între liniile ce conțin cuvintele auxiliare REPETĂ şi SFÂRȘITUL REPETĂRII se numeşte **corpul ciclului**.

Algoritmii ce conțin secvențe de instrucțiuni care în procesul execuției se îndeplinesc de mai multe ori se numesc *algoritmi repetitivi*.

Schemele logice care reprezintă în mod grafic procesele de execuție a procedurii Pătrat şi a programului Opt_pătrate sunt prezentate în *figura 2.6* (p. 76). În aceste scheme se utilizează simbolul grafic REPETĂ, din care, spre deosebire de simbolurile grafice studiate anterior, pleacă două linii orientate: prima spre instrucțiunile din corpul ciclului, iar a doua – spre instrucțiunea care va fi executată imediat după terminarea ciclului.

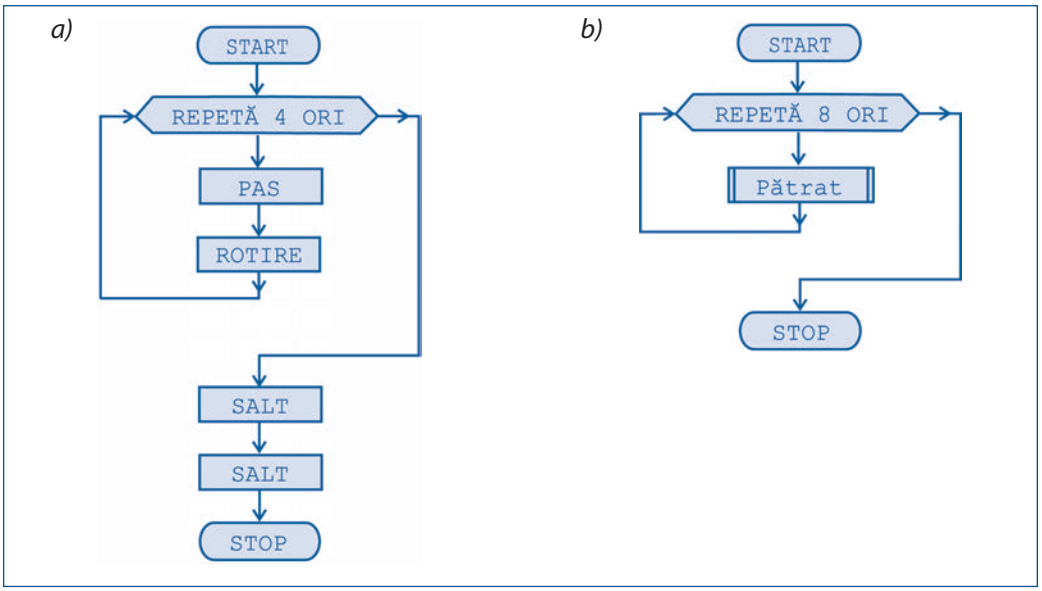


Fig. 2.6. Schemele logice ale algoritmilor repetitivi:
a – procedura `Pătrat`; *b* – programul `Opt_pătrate`

Din analiza schemelor logice ale algoritmilor repetitivi se observă că drumul imaginar parcurs de la simbolul grafic `START` până la simbolul grafic `STOP` reprezintă o linie ce conține cel puțin o buclă. Această buclă include simbolul grafic `REPETĂ` și toate simbolurile grafice ce corespund instrucțiunilor din corpul ciclului (*fig. 2.6*).

Fiind o instrucțiune compusă, instrucțiunea `REPETĂ` poate include în corpul său alte instrucțiuni de acest tip, formându-se astfel o structură imbricată. În consecință, programele ce conțin un număr relativ mic de instrucțiuni pot descrie succesiuni foarte lungi de acțiuni. Pentru exemplificare, prezentăm în continuare un program care impune Cangurul să se deplaseze de 100 de ori de-a lungul marginii de sus a zonei de desenare:

```
ÎNCEPUT
REPETĂ 100 ORI
REPETĂ 15 ORI
SALT
SFÂRŞITUL REPETĂRII
ROTIRE
ROTIRE
SFÂRŞITUL REPETĂRII
SFÂRŞIT
```

Întrebări şi exerciții

❶ Explicați semnificația următoarelor simboluri grafice:

a) START
b) STOP
c) *Instrucțiune*
d) *Nume*
e) REPETĂ *n* ORI
f) →

Care este semnificația liniilor orientate din componența schemelor logice?

❷ **CREEAZĂ!** Desenați schemele logice ale algoritmilor elaborați pentru desenarea figurilor din exercițiile 4, 5 și 8 din paragraful 2.2.

❸ Indicați pe schemele logice din *figura 2.4* drumul imaginar ce simbolizează procesul de execuție a algoritmilor respectivi.

❹ Explicați termenul *algoritm liniar*. Dați exemple.

❺ Care este formatul instrucțiunii `REPETĂ`. Dați câteva exemple de scriere a acestei instrucțiuni.

❻ Când este rezonabilă folosirea instrucțiunii `REPETĂ`? Dați exemple.

❼ Explicați termenul *algoritm repetitiv*. Dați exemple.

❽ Indicați pe schemele logice din *figura 2.6* drumul imaginar ce simbolizează procesul de execuție a algoritmilor respectivi.

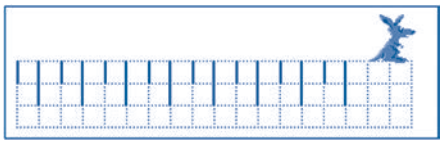
❾ Indicați pe schemele logice din *figura 2.6* buclele care simbolizează ciclurile cu contor.

❿ **STUDIU DE CAZ!** Enumerați instrucțiunile simple pe care le cunoașteți. Prin ce se deosebește o instrucțiune simplă de o instrucțiune compusă?

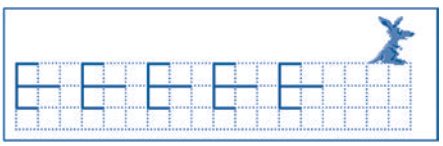
⓫ Elaborați algoritmi repetitivi pentru desenarea figurilor ce urmează:

a)

b)



c)



d)

Calculați câte comenzi va executa Cangurul în procesul desenării acestor figuri.

⓬ **ANALIZEAZĂ!** Câte comenzi va executa Cangurul în procesul derulării următoarelor programe:

a)

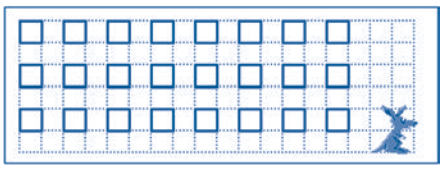
```
ÎNCEPUT
REPETĂ 10 ORI
SALT
ROTIRE
ROTIRE
SALT
SFÂRȘITUL REPETĂRII
SFÂRȘIT
```

b)

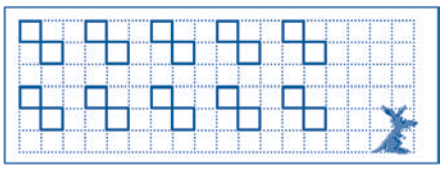
```
ÎNCEPUT
REPETĂ 10 ORI
REPETĂ 20 ORI
SALT
ROTIRE
ROTIRE
SALT
SFÂRȘITUL REPETĂRII
SFÂRȘITUL REPETĂRII
SFÂRȘIT
```

⓭ Utilizând ciclurile imbricate, elaborați algoritmi repetitivi pentru desenarea figurilor ce urmează:

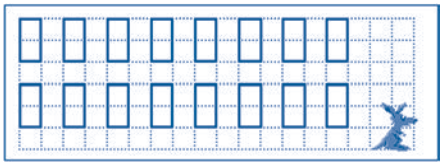
a)



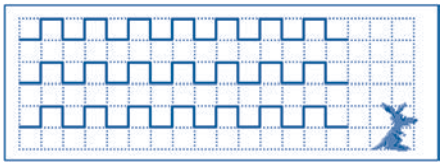
b)



c)



d)



Calculați câte comenzi va executa Cangurul în procesul desenării acestor figuri.

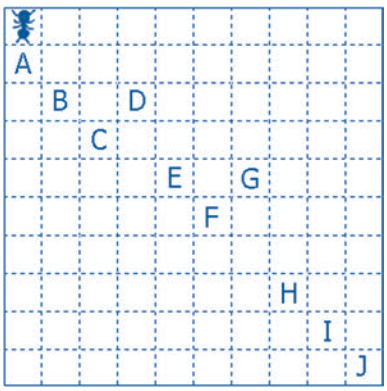
⓮ Scrieți în limbajul executantului **Furnica** următorii subalgoritmi repetitivi:

`Stânga_8` – deplasarea Furnicii în stânga cu opt poziții;
`Dreapta_8` – deplasarea Furnicii în dreapta cu opt poziții;
`Sus_8` – deplasarea Furnicii în sus cu opt poziții;
`Jos_8` – deplasarea Furnicii în jos cu opt poziții.

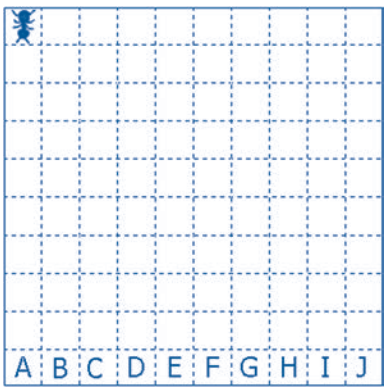
Utilizând acești subalgoritmi, elaborați un program care deplasează Furnica în așa mod, încât traiectoria mișcării să reprezinte un pătrat cu laturile: *a)* 8 × 8; *b)* 9 × 9; *c)* 10 × 10.

⓯ În figurile ce urmează este prezentat aspectul inițial și aspectul final al câmpului de lucru al executantului **Furnica**. Utilizând ciclurile cu contor, scrieți o procedură care mută caracterul imprimabil dintr-o anumită coloană în ultima celulă a coloanei.

Aspectul inițial



Aspectul final



Utilizând ciclurile cu contor, scrieți un program care mută toate caracterele imprimabile în ultima celulă a rândului.

2.4. Algoritmi repetitivi. Ciclu cu condiție

Termeni-cheie:
- senzor
- condiție
- ciclu cu condiție
- algoritm cu conexiune inversă
- eroare de execuție

Până în prezent, toți algoritmii elaborați de noi pentru comanda executanților **Cangurul** și **Furnica** nu analizau situația din mediul de lucru al executanților.

Cu alte cuvinte, Centrul de comandă dirija acțiunile executanților fără să verifice dacă comenzile respective garantează realizarea scopului propus sau dacă executanții sunt în stare să îndeplinească comenzile primite. Acest mod de elaborare a algoritmilor complică esențial soluționarea multor probleme întâlnite frecvent în practica cotidiană.

De exemplu, presupunem că poziția inițială a Furnicii este necunoscută. Se cere să elaborăm un program care deplasează Furnica în colțul stânga-sus al mediului de lucru. Evident, instrucțiunile studiate până acum nu sunt suficiente pentru a soluționa această problemă, întrucât nu cunoaştem câte comenzi de tipul `SUS`, `STÂNGA` trebuie să fie date executantului.

Pentru a evita astfel de situații, executanții sunt dotați cu **senzori***, care comunică Centrului de comandă anumite informații din mediul de lucru. De exemplu, executanții **Cangurul** şi **Furnica** sunt dotați cu senzori care descoperă prezența obstacolelor în direcția unor eventuale deplasări. Informațiile respective sunt transmise Centrului de comandă prin intermediul variabilelor logice `E_LINIE` şi `E_MARGINE`, care pot lua valorile FALS sau `ADEVĂRAT`.

În cazul algoritmilor repetitivi, analiza anumitor situații din mediul de lucru se realizează cu ajutorul instrucțiunii compuse `CÂT`. Această instrucțiune asigură execuția ciclică a unei secvențe de instrucțiuni atâta timp, cât în mediul de lucru se respectă anumite condiții. Formatul şi schema logică a instrucțiunii `CÂT` sunt date în *figura 2.7*.

```
CÂT Condiție
Instrucțiunea_1
Instrucțiunea_2
...
Instrucțiunea_k
SFÂRŞITUL CICLULUI
```

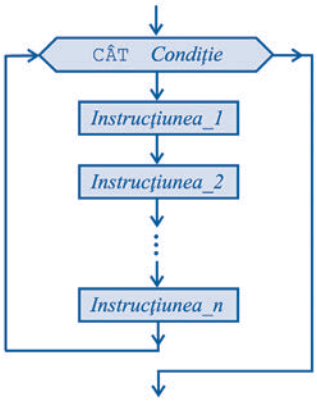


Fig. 2.7. Formatul şi schema logică a instrucțiunii `CÂT`

În descrierea instrucțiunii `CÂT` *Condiție* este o expresie logică, iar cuvintele `CÂT` şi `SFÂRŞITUL CICLULUI` sunt cuvinte auxiliare.

| *Condițiile* reprezintă expresii logice care indică anumite situații din mediul de lucru al executantului.

În limbajele de programare a executanților **Cangurul** şi **Furnica**, expresiile logice utilizate pentru a indica anumite situații din mediul de lucru sunt foarte simple:

```
E_LINIE
E_MARGINE
NU E_LINIE
NU E_MARGINE
```

* Senzor – dispozitiv care sesizează (observă, descoperă) un anumit fenomen.

În cazul unor executanți mai sofisticați, în componența condițiilor pot intra variabile care comunică Centrului de comandă informații despre direcția și viteza mișcării altor obiecte, prezența obstacolelor, temperaturii, umidității, nivelului de radiație etc., iar expresiile logice pot fi formate utilizând operațiile relaționale =, ≠, >, ≥, <, ≤ și operațiile logice NU, ȘI, SAU.

Pentru exemplificare, vom examina următoarea problemă. Presupunem că se dorește deplasarea Cangurului spre una dintre marginile câmpului de lucru, însă poziția inițială a acestuia nu este cunoscută. Evident, în astfel de cazuri utilizatorul nu poate folosi instrucțiunea REPETĂ, întrucât nu se cunoaște numărul de pași pe care trebuie sa-i facă Cangurul. Cu ajutorul instrucțiunii CÂT, astfel de probleme se soluționează foarte ușor:

```
CÂT NU E_MARGINE
SALT
SFÂRȘITUL CICLULUI
```

În procesul execuției instrucțiunii CÂT, Centrul de comandă va analiza mai întâi condiția NU E_MARGINE. Dacă această condiție are valoarea ADEVĂRAT, se execută instrucțiunile din corpul ciclului, în cazul exemplului de mai sus – instrucțiunea SALT. După îndeplinirea instrucțiunii SALT, senzorul din dotarea Cangurului explorează pătrățelul vecin și actualizează valoarea variabilei E_MARGINE. Dacă condiția NU E_MARGINE ia valoarea FALS, execuția ciclului se termină și se trece la instrucțiunea ce urmează imediat după cuvintele auxiliare SFÂRȘITUL CICLULUI.

Instrucțiunea CÂT se numește **ciclu cu condiție**, deoarece la execuția ei se repetă ciclic aceeași secvență de instrucțiuni, iar numărul de repetări se stabilește în procesul derulării programului în funcție de valorile curente ale condiției respective.

Implementarea instrucțiunii CÂT presupune existența dintre Centrul de comandă și executant a două canale de transmitere a informației:

1) canalul direct, destinat transmiterii comenzilor de la Centrul de comandă la executant;

2) canalul invers, destinat transmiterii informațiilor colectate cu ajutorul senzorilor de la executant la Centrul de comandă.

Algoritmii ce conțin secvențe de instrucțiuni, execuția cărora depinde de informațiile colectate în mediul de lucru al executanților, se numesc *algoritmi cu conexiune inversă*.

Majoritatea algoritmilor utilizați în informatica modernă sunt algoritmi cu conexiune inversă, fapt ce permite executanților să se adapteze la situațiile concrete din mediul de lucru. Pentru exemplificare, prezentăm în continuare un program foarte mic, care desenează o figură destul de complicată – o spirală (*fig. 2.8*):

```
ÎNCEPUT
CÂT NU E_LINIE
CÂT NU E_LINIE
PAS
SFÂRȘITUL CICLULUI
ROTIRE
SFÂRȘITUL CICLULUI
SFÂRȘIT
```

Ciclul interior, corpul căruia conține instrucțiunea PAS, asigură desenarea unui segment de dreaptă, lungimea căruia depinde de configurația liniilor trasate anterior. Imediat cum senzorul Cangurului depistează în pătrățelul vecin o linie, variabila NU_E_LINIE obține valoarea FALS și execuția acestui ciclu se termină. Instrucțiunea imediat următoare rotește Cangurul cu 90°, pregătindu-l astfel pentru trasarea unei linii noi. Ciclul exterior asigură trasarea liniilor ce formează spirala până când se ajunge la situația în care toate pătrățelele vecine sunt deja ocupate.

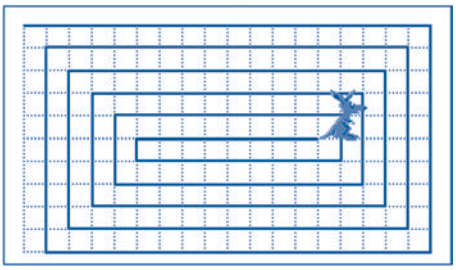


Fig. 2.8. Spirala desenată de Cangur

Teoretic, o spirală ar putea fi trasată și fără utilizarea instrucțiunii CÂT, folosind pentru organizarea ciclurilor instrucțiunea REPETĂ. Însă în acest caz, utilizatorul va fi nevoit să calculeze în mod manual lungimea fiecărei linii din componența spiralei și să scrie câte patru cicluri cu contor pentru fiecare spiră. Evident, în cazul unui număr mare de spire, acest lucru, practic, este imposibil.

Din practică s-a stabilit că în procesul elaborării și depanării programelor apar situații când executantul primește comenzi pe care nu le poate îndeplini. De exemplu, programul

```
ÎNCEPUT
REPETĂ 1000 ORI
PAS
SFÂRȘITUL REPETĂRII
SFÂRȘIT
```

este corect din punct de vedere sintactic, însă derularea lui completă este imposibilă, întrucât dimensiunile câmpului de lucru sunt mult mai mici de 1000 × 1000. După lansarea în execuție a acestui program, indiferent de poziția inițială, se va ajunge în situația în care comanda PAS nu mai poate fi îndeplinită, întrucât Cangurul se află deja la marginea câmpului. Prin urmare, în procesul derulării programului de mai sus apare o eroare de execuție sau un refuz.

> ***Eroarea de execuție* (refuzul) apare atunci când în procesul derulării unui program executantul nu poate îndeplini comanda primită.**

Reacția Centrului de comandă la o eroare de execuție depinde de tipul executantului și posibilitățile programului respectiv. De exemplu, executanții **Cangurul** și **Furnica** afișează un mesaj de eroare și opresc derularea programului. În cazul unor executanți mai sofisticați, refuzul este tratat ca o condiție specială, care declanșează execuția unor subalgoritmi de corectare a erorilor apărute. Pentru exemplificare, amintim sistemele de asistență a editoarelor de texte și a aplicațiilor de calcul tabelar, care, în cazul unei erori, afișează pe ecran sugestii ce îl ajută pe utilizator să lanseze comenzile respective corecte.

Întrebări şi exerciţii

❶ Care este destinaţia senzorilor cu care sunt dotaţi executanţii?

❷ Pentru ce se utilizează condiţiile? Cine atribuie valori acestor variabile?

❸ **CREEAZĂ!** Reprezentaţi pe un desen fluxul de informaţii dintre Centrul de comandă şi executant. Care este destinaţia canalului direct şi a canalului invers?

❹ **STUDIU DE CAZ!** Prin ce se deosebesc algoritmii cu conexiune inversă de algoritmii fără astfel de conexiuni?

❺ Ar putea oare calculatorul să conducă un automobil fără ca să utilizeze algoritmi cu conexiune inversă? Argumentaţi răspunsul dvs.

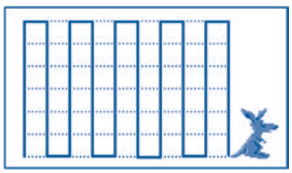
❻ **ANALIZEAZĂ!** Creaţi schema logică a programului pentru desenarea unei spirale. Indicaţi pe această schemă drumul imaginar ce simbolizează procesul de execuţie a programului şi buclele care corespund ciclurilor cu condiţie.

❼ **EXPLOREAZĂ!** Încercaţi să elaboraţi un program care ar desena o spirală fără utilizarea ciclului cu condiţie. Estimaţi numărul de instrucţiuni ale unui astfel de program.

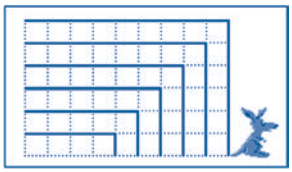
❽ Utilizând ciclul cu condiţie, elaboraţi programe pentru desenarea figurilor de mai jos.

a)

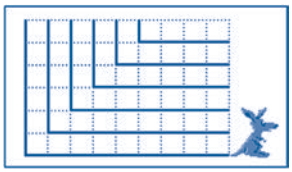
b)



c)



d)



Desenaţi schemele logice ale programelor elaborate. Indicaţi pe fiecare schemă drumul imaginar ce simbolizează procesul de execuţie a programului şi buclele care corespund ciclurilor cu condiţie.

❾ Elaboraţi un program care, fără să cunoască poziţia iniţială, deplasează Furnica în una dintre următoarele poziţii:
a) colţul stânga-sus; *b)* colţul dreapta-sus; *c)* colţul stânga-jos; *d)* colţul dreapta-jos.
Se consideră că toate pătrăţelele mediului de lucru sunt libere.

❿ Elaboraţi un program care deplasează necontenit Furnica în aşa mod, încât traiectoria mişcării ei să reprezinte un pătrat, laturile căruia coincid cu marginile mediului de lucru. Iniţial Furnica se află în colţul stânga-sus al mediului de lucru.

2.5. Algoritmi cu ramificări

Termeni-cheie:
- ramificator
- algoritm cu ramificări

Foarte des, în procesul execuţiei unui program, apare necesitatea de a da anumite comenzi executantului în funcţie de situaţia din mediul de lucru. Cunoaştem deja

că informațiile referitoare la situația din mediul de lucru sunt colectate cu ajutorul senzorilor, datele respective fiind reprezentate în program prin variabile de genul E_LINIE, E_MARGINE etc. În paragraful precedent am folosit astfel de variabile pentru a scrie condițiile instrucțiunilor compuse CÂT.

În general, condițiile care descriu situația din mediul de lucru pot fi analizate şi cu ajutorul altei instrucțiuni compuse – instrucțiunea DACĂ. Formatul şi schema logică a instrucțiunii DACĂ sunt date în *figura 2.9*.

```
DACĂ Condiție
ATUNCI Instrucțiunea_1
ALTFEL Instrucțiunea_2
```

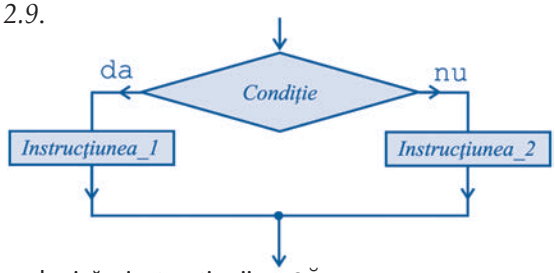


Fig. 2.9. Formatul şi schema logică a instrucțiunii DACĂ

În descrierea instrucțiunii DACĂ *Condiție* este o expresie logică, iar cuvintele DACĂ, ATUNCI şi ALTFEL sunt cuvinte auxiliare.

Ca şi în cazul instrucțiunii CÂT, în limbajele de programare a executanților **Cangurul** şi **Furnica** putem utiliza următoarele expresii logice:

```
E_LINIE
E_MARGINE
NU E_LINIE
NU E_MARGINE
```

Evident, în cazul unor executanți mai sofisticați, în componența condițiilor pot intra şi alte variabile, iar expresiile logice pot fi formate utilizând operațiile relaționale =, ≠, >, ≥, <, ≤ şi operațiile logice NU, ŞI, SAU.

În procesul execuției instrucțiunii DACĂ, Centrul de comandă va analiza mai întâi condiția respectivă. Dacă această condiție are valoarea ADEVĂRAT, se execută *Instrucțiunea_1*, iar în caz contrar (condiția are valoarea FALS), se execută *Instrucțiunea_2*.

Instrucțiunea compusă DACĂ se numeşte **ramificator**, deoarece drumul imaginar, care simbolizează procesul de execuție, va trece, în funcție de valorile curente ale condiției analizate, prin simbolul grafic *Instrucțiunea_1* sau prin simbolul grafic *Instrucțiunea_2*, rombul reprezentând punctul de ramificare.

Algoritmii ce conțin secvențe de instrucțiuni care vor fi executate numai pentru anumite valori ale condițiilor indicate se numesc *algoritmi cu ramificări*.

Pentru exemplificare, prezentăm un program care desenează un pătrat, laturile căruia coincid cu marginile câmpului de lucru:

```
ÎNCEPUT
REPETĂ 100 ORI
DACĂ E_MARGINE
ATUNCI ROTIRE
ALTFEL PAS
SFÂRŞITUL REPETĂRII
SFÂRŞIT
```

Corpul ciclului REPETĂ al acestui program conține instrucțiunea compusă DACĂ, care va fi executată de 100 de ori. La fiecare execuție se verifică condiția E_MARGINE și, în funcție de valoarea curentă, se execută instrucțiunea ROTIRE sau instrucțiunea PAS. Evident, instrucțiunea ROTIRE se va executa numai atunci când valoarea condiției va fi ADEVĂRAT sau, cu alte cuvinte, când Cangurul va ajunge la marginea câmpului de lucru.

Schema logică a programului dat este prezentată în *figura 2.10*. Din această schemă se observă că la fiecare execuție repetată a instrucțiunii DACĂ, alegerea uneia dintre instrucțiunile ROTIRE sau PAS se va efectua în funcție de poziția curentă a Cangurului față de marginea câmpului de lucru.

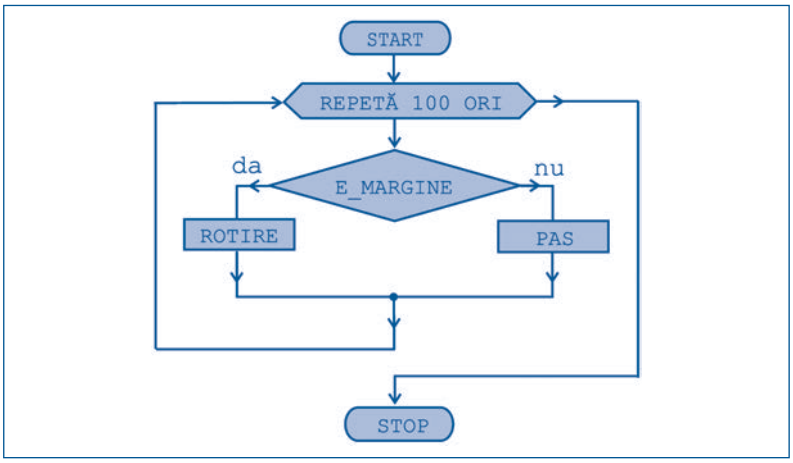


Fig. 2.10. Schema logică a algoritmului cu ramificări

Întrebări și exerciții

❶ Care este destinația instrucțiunii DACĂ? Scrieți formatul și desenați schema logică a acestei instrucțiuni.

❷ Cum se execută instrucțiunea DACĂ? Poate fi oare înlocuită instrucțiunea DACĂ cu instrucțiunea CÂT?

❸ Este oare, în opinia dvs., algoritmul cu ramificări un algoritm cu conexiune inversă? Argumentați răspunsul.

❹ **EXPLOREAZĂ!** Numiți cel puțin trei probleme din matematică, algoritmii de soluționare a cărora utilizează ramificatori.

❺ Care este diferența dintre algoritmii liniari și algoritmii cu ramificări?

❻ Care este diferența dintre algoritmii repetitivi și algoritmii cu ramificări?

❼ **STUDIU DE CAZ!** Elaborați două programe care desenează o spirală în interiorul unui dreptunghi, dimensiunile căruia nu sunt cunoscute (*fig. 2.11*). Pentru a analiza situația de pe câmpul de lucru, în primul program se va utiliza instrucțiunea DACĂ, iar în al doilea – instrucțiunea CÂT. Care program este mai eficient, cel ce utilizează instrucțiunea DACĂ sau cel ce utilizează instrucțiunea CÂT? Argumentați răspunsul dvs.

❽ **CREEAZĂ!** Desenați schemele logice ale programelor destinate trasării spiralei în interiorul unui dreptunghi (*fig. 2.11*). Indicați pe aceste scheme drumurile imaginare ce simbolizează procesele de execuție și punctul de ramificare.

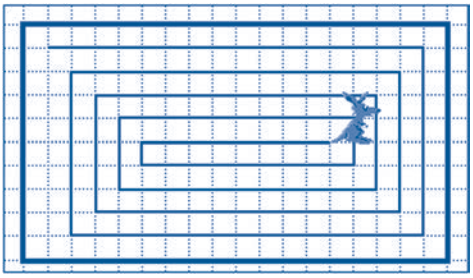


Fig. 2.11. Spirala desenată în interiorul unui dreptunghi

❾ **ELABOREAZĂ!** Cangurul se află la începutul unui coridor, lățimea căruia este de două pătrățele (*fig. 2.12*). Elaborați un program care desenează o linie ce va trece exact prin mijlocul coridorului. Se admite că la momentul scrierii programului, dimensiunile coridorului, cu excepția lățimii, nu sunt cunoscute.

a) *b)*

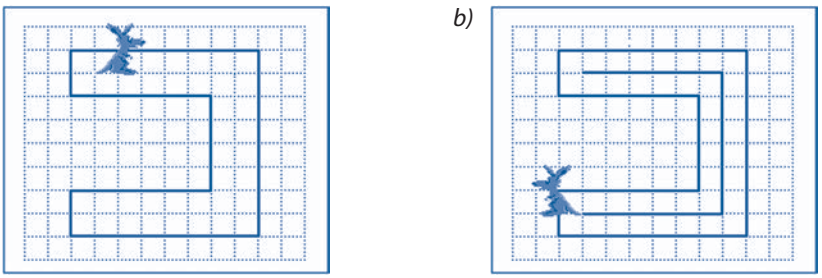


Fig. 2.12. Trasarea liniilor în interiorul unui coridor:
a – poziția inițială; *b* – poziția finală

2.6. Algoritmul de funcţionare a calculatorului

Termeni-cheie:
- unități funcționale
- dispozitivul central de comandă
- dispozitivul aritmetic şi logic
- comenzi frecvent utilizate
- algoritmul de funcționare

Un calculator numeric conține următoarele **unități funcționale** (*fig. 2.13*, p. 86): procesorul, memoria internă, memoria externă, dispozitivul de intrare şi dispozitivul de ieşire. La rândul său, procesorul este format din **dispozitivul central de comandă** şi **dispozitivul aritmetic şi logic**.

Cu excepția dispozitivului central de comandă, toate unitățile funcționale ale calculatorului pot fi tratate ca executanți ce îndeplinesc anumite comenzi. Pentru exemplificare, în *tabelul 2.1* (p. 86) sunt prezentate **comenzile frecvent utilizate** ale executanților din componența unui calculator personal.

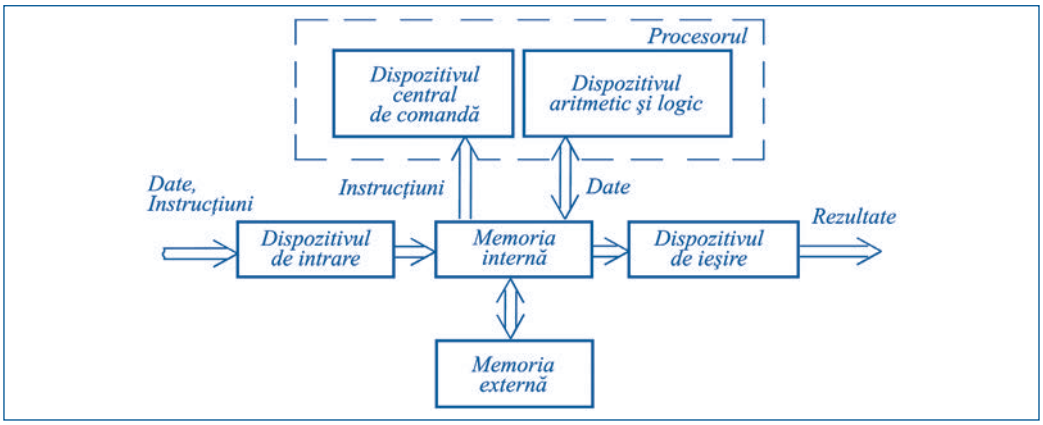


Fig. 2.13. Schema funcţională a calculatorului

Tabelul 2.1

Comenzile frecvent utilizate ale executanţilor din componenţa unui calculator personal

Nr. crt.	Executantul	Comenzile executantului
1.	Tastatura	Aşteaptă acţionarea unei taste Citeşte codul tastei apăsate Blochează tastatura
2.	Monitorul	Şterge ecranul Afişează caracterul indicat Afişează elementul grafic indicat Stabileşte culoarea de fundal
3.	Unitatea de disc magnetic	Înscrie datele pe disc Citeşte datele de pe disc
4.	Imprimanta	Tipăreşte caracterul indicat Avans de linie Avans de pagină
5.	Unitatea de disc optic	Citeşte datele de pe disc Extrage discul
6.	Memoria internă	Citeşte datele din locaţia indicată Înscrie datele în locaţia indicată
7.	Dispozitivul aritmetic şi logic	Adună Scade Înmulţeşte Împarte Compară ŞI SAU NU

Dispozitivul central de comandă asigură dirijarea executanţilor conform programului înscris în memoria internă a calculatorului. Programul constă dintr-un set de instrucţiuni, codificate în formă de cuvinte binare, în care se indică operaţia ce

trebuie executată şi amplasamentul (locul) operanzilor. De exemplu, într-o instrucțiune aritmetică se indică operația ce trebuie efectuată (adunarea, scăderea, înmulțirea sau împărțirea) şi amplasamentul operanzilor în memoria internă. Într-o instrucțiune de introducere a informației se indică dispozitivul de intrare (tastatura, cititorul de documente) şi locul din memoria internă unde va fi stocată informația introdusă. Într-o instrucțiune de extragere a informației se indică locul din memoria internă ce conține informația respectivă şi dispozitivul de ieşire (monitorul sau imprimanta). În mod similar, în cazul citirii sau scrierii informației pe un disc magnetic, în instrucțiunea respectivă se indică amplasamentul informației în memoria internă şi unitatea de disc.

Algoritmul de funcționare a dispozitivului central de comandă şi, implicit, a calculatorului în ansamblu, poate fi descris în felul următor:

```
CÂT INSTRUCȚIUNEA EXTRASĂ DIN MEMORIA INTERNĂ ≠ STOP
EXTRAGE O INSTRUCȚIUNE DIN MEMORIA INTERNĂ
DECODIFICĂ INSTRUCȚIUNEA EXTRASĂ
TRANSMITE COMENZI EXECUTANȚILOR
SFÂRŞITUL CICLULUI
```

Din algoritmul prezentat observăm că dispozitivul central de comandă realizează principiul de comandă prin program, transmițând executanților din componența calculatorului comenzi, generate în baza instrucțiunilor, extrase din memoria internă. Accentuăm faptul că repertoriul de instrucțiuni al unui calculator modern include atât instrucțiuni pentru prelucrarea informației (adunarea, scăderea, înmulțirea, împărțirea etc.), cât şi instrucțiuni pentru apelurile de subprograme, realizarea algoritmilor repetitivi şi a algoritmilor cu ramificații. Un astfel de repertoriu permite descrierea compactă a unor prelucrări foarte complexe, care, împreună cu vitezele de operare foarte mari (milioane de instrucțiuni pe secundă), asigură aplicarea eficientă a calculatoarelor în toate domeniile ştiinței şi tehnicii moderne.

Întrebări şi exerciții

❶ **CERCETEAZĂ!** Determinați repertoriul de comenzi al fiecărei unități periferice (tastatura, imprimanta, monitorul, cititorul de documente etc.) din laboratorul de informatică.

❷ **EXPLOREAZĂ!** Utilizând descrierile tehnice ale calculatoarelor din laboratorul de informatică, determinați repertoriul de instrucțiuni al calculatorului la care lucrați dvs. Indicați instrucțiunile de prelucrare a informației, instrucțiunile de intrare–ieşire şi instrucțiunile de control.

❸ Explicați destinația fiecărei unități funcționale din componența unui calculator (*fig. 2.13*).

❹ Care este rolul dispozitivului central de comandă din componența procesorului?

❺ Formulați algoritmul de funcționare a unui calculator numeric. Enumerați comenzile pe care le transmite dispozitivul central de comandă executanților din componența unui calculator.

❻ **ANALIZEAZĂ!** De ce depinde capacitatea de prelucrare a unui calculator? Argumentați răspunsul dvs.

❼ Ce operații efectuează dispozitivul central de comandă în procesul executării unui program? Cum poate fi întreruptă derularea unui program?

❽ Cum se realizează principiul de comandă prin program în cazul unui calculator numeric?

CERCETEAZĂ! Cum credeți, care este destinația calculatoarelor instalate pe automobilele moderne? Ce fel de unități periferice are un astfel de calculator? Ce fel de programe derulează pe aceste calculatoare?

2.7. Generalităţi despre algoritmi

Termeni-cheie:
- reprezentarea algoritmilor
- proprietățile algoritmilor
- clasificarea algoritmilor
- gândirea algoritmică
- cultura informațională

Pentru a simplifica procesele de elaborare a algoritmilor, s-a convenit ca aceştia să fie reprezentați (descrişi, notați) cu ajutorul unor mijloace standard, cele mai răspândite fiind:

1) limbajul de comunicare între oameni, cu utilizarea, în caz de necesitate, a formulelor matematice;

2) schemele logice;

3) limbajele algoritmice (limbajele de programare).

De exemplu, algoritmul de soluționare a ecuațiilor de gradul I $ax + b = 0$ poate fi descris în limba română în felul următor:

1. Citeşte de la tastatură coeficienții a şi b.
2. Dacă $a \neq 0$, atunci calculează rădăcina $x = -b / a$, afişează pe ecran mesajul "Ecuația are o singură rădăcină" şi valoarea x. Stop.
3. Dacă $a = 0$ şi $b = 0$, afişează pe ecran mesajul "Ecuația are o mulțime infinită de rădăcini". Stop.
4. Dacă $a = 0$ şi $b \neq 0$, atunci afişează pe ecran mesajul "Ecuația nu are sens". Stop.

Avantajul principal al modului de descriere a algoritmilor cu ajutorul unui limbaj de comunicare între oameni constă în faptul că descrierile respective sunt înțelese de orice persoană care cunoaşte instrucțiunile şi operațiile ce apar în textul algoritmului. Cu regret, calculatoarele moderne nu pot interpreta univoc limbajele vorbite de oameni, iar algoritmii descrişi în acest limbaj nu pot fi utilizați ca programe de calculator.

În cazul schemelor logice, algoritmii sunt descrişi cu ajutorul simbolurilor grafice `START`, `STOP`, *Instrucțiune*, *Apel de subalgoritm*, `REPETĂ`, `CÂT`, `DACĂ` etc. Pentru exemplificare, în *figura 2.14* este prezentată schema logică a algoritmului pentru soluționarea ecuațiilor de gradul I.

Analiza schemelor logice ne permite să efectuăm mai uşor **clasificarea algoritmilor** în algoritmi liniari, algoritmi repetitivi şi algoritmi cu ramificații. Schemele logice sunt foarte sugestive, însă, ca şi în cazul limbajelor de comunicare între oameni, descrierile respective nu pot fi interpretate univoc de calculatoarele moderne. Prin urmare, schemele logice nu pot fi utilizate ca programe de calculator.

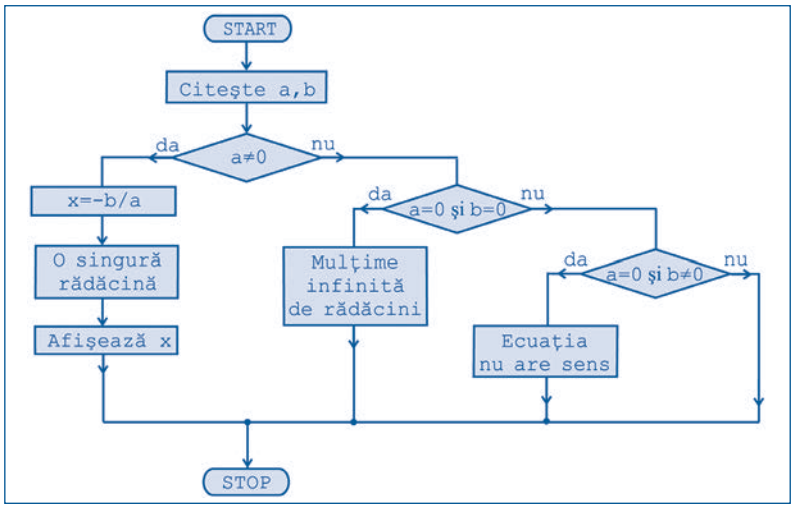


Fig. 2.14. Schema logică a algoritmului pentru soluționarea ecuațiilor de gradul I

Limbajele algoritmice reprezintă limbaje artificiale, create în scopul descrierii exacte a algoritmilor prin utilizarea unui vocabular, a unei sintaxe şi a unei semantici bine definite. Pentru exemplificare, prezentăm în continuare algoritmul de soluționare a ecuațiilor de gradul I, scris în limbajul algoritmic PASCAL:

```
Program Exemplu;
var a, b, x : real;
begin
  readln(a, b);
  if a<>0 then
    begin
     x:=-b/a;
      writeln('Ecuatia are o singura radacina');
     writeln(x);
    end;
  if (a=0) and (b=0) then
     writeln('Ecuatia are o multime infinita de radacini');
  if (a=0) and (b<>0) then
     writeln('Ecuatia nu are sens');
end.
```

Cuvintele engleze din această descriere au următoarea semnificație: **Program** – program; **var** – variabilă; **begin** – început; **end** – sfârşit; **if** – dacă; **then** – atunci; `readln` – citeşte; `writeln` – scrie; **and** – şi.

Constatăm că limbajul algoritmic PASCAL conține instrucțiuni şi cuvinte auxiliare, pe care le-am studiat în paragrafele precedente în procesul elaborării programelor pentru comanda executanților **Cangurul** şi **Furnica**.

Avantajul principal al limbajelor algoritmice constă în faptul că ele asigură o descriere univocă a algoritmilor, sunt înțelese de oameni şi pot fi traduse relativ uşor în limbajul executanților. Evident, studierea unui limbaj algoritmic, ca şi studierea unei

limbi noi, necesită un anumit efort din partea utilizatorului. În şcolile din ţara noastră, limbajul algoritmic PASCAL se studiază începând cu clasa a 10-a.

Limbajele de programare reprezintă limbaje algoritmice, adaptate pentru descrierea algoritmilor într-o formă înţeleasă de executanţi. De exemplu, programele elaborate în paragrafele precedente au fost scrise în limbajul executanţilor **Cangurul** şi **Furnica**. Acest limbaj de programare conţine, în afară de instrucţiunile `REPETĂ`, `CÂT`, `DACĂ`, apel de procedură etc., întâlnite în toate limbajele algoritmice, şi instrucţiuni specifice anume acestor executanţi: PAS, `SALT`, `ROTIRE`, `SUS`, JOS etc.

Indiferent de forma în care sunt reprezentaţi algoritmii, ei dispun de un şir de proprietăţi comune, care le deosebesc de instrucţiunile, recomandările sau planurile elaborate pentru soluţionarea unor probleme concrete. S-a stabilit că orice algoritm are trei proprietăţi distincte:

1) **determinismul** – cunoaşterea cu exactitate în fiecare moment al execuţiei algoritmului a următoarei operaţii de executat, precum şi modul de execuţie a fiecărei operaţii;

2) **universalitatea** – algoritmul este aplicabil pentru soluţionarea tuturor problemelor pentru care a fost elaborat;

3) **finitudinea** – algoritmul este finit în spaţiu (ca descriere) şi timp (ca execuţie).

De obicei, în procesul elaborării, utilizatorul schiţează algoritmul într-un limbaj de comunicare între oameni, de exemplu în limba română. Pe parcurs, pentru o descriere sugestivă a prelucrărilor preconizate, pot fi utilizate şi schemele logice. Evident, algoritmul în versiunea finală va fi scris într-un limbaj algoritmic sau, în cazul unui executant concret, direct în limbajul respectiv de programare.

În general, elaborarea algoritmilor reprezintă un proces de creaţie care presupune că utilizatorul posedă aşa-numita gândire algoritmică.

Prin *gândire algoritmică* înţelegem capacitatea persoanei de a elabora algoritmi pentru soluţionarea problemelor pe care ea le întâlneşte în viaţa cotidiană.

Desigur, gândirea algoritmică se formează şi se dezvoltă în procesul studierii mai multor discipline şcolare, rolul central revenindu-i informaticii. Să reţinem că dacă un secol în urmă sarcina principală a instituţiilor de învăţământ consta în cultivarea ştiinţei de carte, astăzi această sarcină s-a extins, incluzând formarea **culturii informaţionale** şi dezvoltarea gândirii algoritmice.

Întrebări şi exerciţii

❶ **STUDIU DE CAZ!** Care sunt mijloacele principale de reprezentare a algoritmilor? Care sunt avantajele şi dezavantajele fiecărui mod de notare a algoritmilor?

❷ Încercaţi să descrieţi procesul de elaborare a unui algoritm. Ce fel de mijloace se folosesc pentru a descrie un algoritm în curs de elaborare?

❸ Există oare vreo diferenţă între un limbaj algoritmic şi un limbaj de programare? Argumentaţi răspunsul dvs.

❹ **EXPLOREAZĂ!** Explicaţi termenii *gândire algoritmică* şi *cultură informaţională*. Utilizând un server de căutare, găsiţi pe Internet informaţii referitoare la aceşti termeni.

❺ **ANALIZEAZĂ!** Enumeraţi proprietăţile de bază ale algoritmilor. Verificaţi dacă programele elaborate pentru comanda executanţilor **Cangurul** şi **Furnica** posedă aceste proprietăţi.

❻ Elaboraţi un algoritm pentru soluţionarea ecuaţiilor de gradul II: $ax^2+bx+c=0$. Reprezentaţi acest algoritm în formă de schemă logică.

Capitolul 3

IMPLEMENTAREA ALGORITMILOR ÎN MEDII TEXTUALE DE PROGRAMARE

3.1. Conceptul de acţiune

Conform **conceptului de acțiune** realizat în limbajul PASCAL, calculatorul reprezintă un executant, mediul de lucru al căruia este format din mulțimea tuturor variabilelor şi constantelor declarate în programul respectiv. În procesul derulării programului, executantul efectuează asupra mărimilor din mediul de lucru anumite acțiuni (operații), de exemplu: adunarea sau scăderea, citirea de la tastatură sau afişarea pe ecran etc. În urma acestor acțiuni valorile variabilelor pot fi schimbate, iar cele ale constantelor – nu.

Operațiile necesare pentru a prelucra datele unui program şi ordinea executării lor se definesc cu ajutorul **instrucțiunilor**. Există două categorii de instrucțiuni:

1) instrucțiuni simple;
2) instrucțiuni structurate.

Instrucțiunile simple nu conțin alte instrucțiuni. Instrucțiunile simple sunt:

– instrucțiunea de atribuire;
– instrucțiunea de apel de procedură;
– instrucțiunea de salt necondiționat;
– instrucțiunea de efect nul.

Instrucțiunile structurate sunt construite din alte instrucțiuni. Instrucțiunile structurate sunt:

– instrucțiunea compusă;
– instrucțiunile condiționale **`if`** şi **`case`**;
– instrucțiunile iterative **`for`**, **`while`** şi **`repeat`**;
– instrucțiunea **`with`**.

Instrucțiunile **`if`** şi **`case`** se utilizează pentru programarea algoritmilor cu ramificări, iar instrucțiunile **`for`**, **`while`** şi **`repeat`** – pentru programarea algoritmilor repetitivi. Amintim că algoritmii repetitivi se folosesc pentru descrierea unor prelucrări care trebuie executate de mai multe ori, iar cei cu ramificări – pentru a selecta prelucrările dorite în funcție de condițiile din mediul de lucru al executantului.

În cadrul unui program instrucțiunile pot fi prefixate de etichete. Etichetele pot fi referite în instrucțiunile de salt necondiționat **`goto`**. Amintim că eticheta este un număr întreg fără semn care se separă de instrucțiunea propriu-zisă prin simbolul ":" (două puncte).

Într-un program PASCAL scrierea instrucțiunilor pe linii nu este limitată, o instrucțiune poate ocupa una sau mai multe linii, sau într-o linie pot fi mai multe instrucțiuni. Ca separator de instrucțiuni se foloseşte simbolul ";" (punct şi virgulă).

3.2. Expresii

Formulele pentru calculul unor valori se reprezintă în PASCAL prin **expresii**. Acestea sunt formate din operanzi (constante, variabile, referințe de funcții) și operatori (simbolurile operațiilor). Operatorii se clasifică după cum urmează:

– operatori multiplicativi: *, /, **div**, **mod**, **and**;
– operatori aditivi: +, –, **or**;
– operatori relaționali: <, <=, =,>=, >, <>, **in**.

În PASCAL expresiile se scriu pe un singur rând, fără a utiliza indici. Amintim că indicii reprezintă simboluri așezate la dreapta sau la stânga (mai sus sau mai jos) față de un număr sau de o literă, cărora le precizează valoarea sau înțelesul. Prin urmare, expresia matematică $\frac{a+b}{3}$ va fi scrisă în PASCAL ca `(a+b)/3`, iar expresia matematică x^2 va fi scrisă în Pascal ca `x*x`.

Exemple:

	Notația matematică	*Notația în PASCAL*
1)	$\frac{a+b}{c+d}$	`(a+b)/(c+d)`
2)	$\frac{-b+\sin(b-c)}{2a}$	`(-b+sin(b-c))/(2*a)`
3)	$-\frac{1}{xy}$	`-1/(x*y)`
4)	$p<q \,\&\, r>s$	`(p<q) and (r>s)`
5)	$\overline{x \vee y}$	`not (x or y)`
6)	$\frac{1}{a+b}>\frac{1}{c+d}$	`1/(a+b)>1/(c+d)`

În cadrul unei expresii, un apel de funcție poate să apară peste tot unde există o variabilă sau o constantă. Limbajul PASCAL conține un set de **funcții predefinite**, cunoscute oricărui program.

Exemple:
`abs(x)` – valoarea absolută ($|x|$);
`sqr(x)` – pătratul lui x (x^2);
`sqrt(x)` – rădăcina pătrată ($\sqrt{x}$).

Lista tuturor funcțiilor predefinite ale limbajului PASCAL poate fi consultată cu ajutorul sistemului de asistență al mediului de programare.

Întrebări și exerciții

❶ Scrieți conform regulilor limbajului PASCAL expresiile:

a) a^2+b^2;

b) $a^2+2ab+b^2$;

c) $(a+b)^2$;

d) $v_0 t + \frac{at^2}{2}$;

e) $\frac{-b+\sqrt{b^2-4ac}}{2a}$;

f) $\cos\alpha + \cos\beta$;

g) $\cos(\alpha+\beta)$;

h) $2\pi r$;

i) πr^2;

j) $x_1x_2 \vee x_3x_4$;

k) $\overline{x_1 \vee x_2}$;

l) $|x| < 3$;

m) $|z| < 6 \;\&\; |q| > 3{,}14$;

n) $x > 0 \;\&\; y > 8 \;\&\; R < 15$.

❷ Transpuneți expresiile PASCAL în notații obișnuite:

a) `sqr(a)+sqr(b)`

b) `2*a*(b+c)`

c) `sqrt((a+b)/(a-b))`

d) `exp(x+y)`

e) `cos(ALFA-BETA)`

f) `sqr(a+b)/(a-b)`

g) `x>0 or q<p`

h) `not(x and y)`

❸ Care dintre expresiile PASCAL ce urmează sunt greșite?

a) `((((+x))))`

b) `((((x))))`

c) `a<<b or c>d`

d) `not q and p`

e) `q and not p`

f) `not q and p`

g) `a+-b`

h) `sinx+cosx`

i) `sqr(x)+sqr(y)`

j) `sin(-x)`

k) `sin-x`

l) `cos(x+y)`

m) `sin(abs(x)+abs(y))`

n) `sqrt(-y)`

3.3. Evaluarea expresiilor

Prin evaluarea unei expresii se înțelege calculul valorii ei. Rezultatul furnizat depinde de valorile operanzilor şi de operatorii care acționează asupra acestora. Regulile de evaluare a unei expresii sunt cele obişnuite în matematică:

– operațiile se efectuează conform priorității operatorilor;
– în cazul priorităților egale, operațiile se efectuează de la stânga spre dreapta;
– mai întâi se calculează expresiile dintre paranteze.

Prioritățile operatorilor sunt indicate în *tabelul 3.1*.

Tabelul 3.1

Prioritățile operatorilor limbajului PASCAL

Categorie	Operatori	Prioritate
operatori unari	`not, @`	prima (cea mai mare)
operatori multiplicativi	`*, /, div, mod, and`	a doua
operatori aditivi	`+, -, or`	a treia
operatori relaționali	`<, <=, =, >=, >, <>, in`	a patra (cea mai mică)

Exemplu:

Fie $x = 2$ şi $y = 6$. Atunci:

1) `2*x+y = 2*2+6 = 4+6 = 10;`

2) `2*(x+y) = 2(2+6) = 2·8 = 16;`

3) `x+y/x-y = 2+6/2-6 = 2+3-6 = 5-6 = -1;`

4) `(x+y)/x-y = (2+6)/2-6 = 8/2-6 = 4-6 = -2;`

5) `x+y/(x-y) = 2+6/(2-6) = 2+6/(-4) = 2+(-1,5) = 0,5;`

6) `x+y<15 = 2+6<15 = 8<15 = true;`

7) `(x+y<15) and (x>3) = (2+6<15) and (2>3) = (8<15) and (2>3) =`
`= true and false = false.`

Valoarea curentă a unei expresii poate fi afişată pe ecran cu ajutorul procedurii `writeln`:

```
writeln(<Expresie>)
```

Programul `P38` afişează pe ecran rezultatele evaluării expresiilor `x*y+z` şi `x+y<z-1.0`. Valorile curente ale variabilelor `x`, `y` şi `z` sunt citite de la tastatură.

```
Program P38;
 { Evaluarea expresiilor }
var x, y, z : real;
begin
 writeln('Introduceti numerele reale x, y, z:');
 readln(x, y, z);
 writeln(x*y+z);
 writeln(x+y<z-1.0);
end.
```

Întrebări şi exerciţii

❶ Fie $x = 1$, $y = 2$ şi $z = 3$. Evaluaţi următoarele expresii:

a) `x+y+2*z`

b) `(x+y+2)*z`

c) `x*y+y*z`

d) `x*y<y*z`

e) `(x>y) or (6*x>y+z)`

f) `x*(y+y)*z`

g) `(x*y+y)*z`

h) `x*(y+y)*z`

i) `not (x+y+z>0)`

j) `not (x+y>0) and not (z<0)`

❷ Care sunt regulile de evaluare a unei expresii PASCAL?

❸ Indicaţi prioritatea fiecărui operator al limbajului PASCAL.

❹ Precizaţi ordinea în care se calculează componentele unei expresii PASCAL.

❺ Elaboraţi un program care evaluează expresiile *c* şi *g* din exerciţiul 1. Valorile curente ale variabilelor reale *x*, *y* şi *z* se citesc de la tastatură.

3.4. Tipul expresiilor

În funcţie de mulţimea valorilor pe care le poate lua, fiecare expresie se asociază cu un anumit tip de date. Conform conceptului de dată realizat în limbajul PASCAL, **tipul expresiei** derivă (rezultă) din tipul operanzilor şi operatorilor care acţionează asupra acestora. Prin urmare, tipul unei expresii poate fi dedus fără a calcula valoarea ei.

Tipul rezultatelor furnizate de operatori este indicat în *tabelul 3.2*.

Tabelul 3.2

Tipul rezultatelor furnizate de operatori

Operator	Tipul operanzilor	Tipul rezultatului
+, –, *	`integer`	`integer`
	unul `integer`, altul `real`	`real`
/	`integer` sau `real`	`real`
`div`	`integer`	`integer`
`mod`	`integer`	`integer`
`not`, **`and`**, **`or`**	`boolean`	`boolean`
<, <=, =, >=, >, <>	tipuri identice	`boolean`
	tipuri compatibile	`boolean`
	unul `integer`, altul `real`	`boolean`

Tipul rezultatelor furnizate de funcțiile predefinite ale limbajului PASCAL poate fi aflat cu ajutorul sistemului de asistență al mediului de programare.

Indiferent de tipul operanzilor, operatorul / (împărțirea) furnizează numai rezultate de tip `real`, iar operatorii relaționali – numai rezultate de tip `boolean`.

Pentru a afla tipul unei expresii, factorii, termenii şi expresiile simple se examinează în ordinea evaluării lor. Tipul fiecărei părți componente se deduce cu ajutorul *tabelului 3.2.*

De exemplu, fie expresia:

```
(x>i) or (6*i<sin(x/y))
```

unde `i` este de tipul `integer`, iar `x` şi `y` de tipul `real`.

Aflăm tipul fiecărei părți componente şi al expresiei în ansamblu în ordinea de evaluare:

1)	`x>I`	`boolean;`
2)	`6*i`	`integer;`
3)	`x/y`	`real;`
4)	`sin(x/y)`	`real;`
5)	`6*i < sin(x/y)`	`boolean;`
6)	`(x>i)or(6*i<sin(x/y))`	`boolean.`

Prin urmare, expresia în studiu este de tip `boolean`.

Întrucât în procesul deducției valorile concrete ale expresiilor în studiu nu se calculează, tipurile subdomeniu se extind asupra tipurilor de bază.

De exemplu, în prezența declarațiilor:

```
type  T1=1..10;   { subdomeniu de integer }
      T2=11..20;  { subdomeniu de integer }
var   i:T1;
      j:T2;
```

avem:

	Expresie	*Tipul expresiei*
1)	`i+j`	`integer;`
2)	`i mod j`	`integer;`
3)	`i/j`	`real;`
4)	`sin(i+j)`	`real;`
5)	`i>j`	`boolean.`

ş.a.m.d.

În funcție de tipul expresiei, distingem:

– expresii aritmetice (`integer` sau `real`);

– expresii ordinale (`integer`, `boolean`, `char`, *enumerare*);
– expresii booleene (`boolean`).

De obicei, expresiile aritmetice se utilizează în calcule (instrucțiunea de atribuire), expresiile ordinale – în instrucțiunile **`case`** și **`for`**, iar expresiile booleene – în instrucțiunile **`if`**, **`repeat`** și **`while`**.

Întrebări și exerciții

❶ Prin ce metodă se află tipul unei expresii PASCAL?

❷ În prezența declarațiilor:

```
var x, y : real;
    i, j : integer;
    p, q : boolean;
    r : char;
    s : (A, B, C, D, E, F, G, H);
```

aflați tipul următoarelor expresii:

a) `i mod 3`
b) `i/3`
c) `i mod 3 > j div 4`
d) `x+y/(x-y)`
e) `not(x<i)`
f) `sin(abs(i)+abs(j))`
g) `sin(abs(x)+abs(y))`
h) `p and (cos(x)<=sin(y))`
i) `sqr(i)-sqr(j)`
j) `sqr(x)-sqr(y)`
k) `trunc(x)+trunc(y)`
l) `chr(i)`
m) `ord(r)`
n) `ord(s)>ord(r)`
o) `pred(E)`
p) `(-x+sin(x-y))/(2*i)`

❸ Tipul unei expresii poate fi aflat din forma textuală a rezultatelor afișate pe ecran de instrucțiunea `writeln` (<*Expresie*>).

Exemple:

	Rezultatul afișat pe ecran	*Tipul expresiei*
1)	`100`	`integer;`
2)	`1.0000000000E+02`	`real;`
3)	`true`	`boolean.`

Elaborați programele respective și precizați tipul expresiilor ce urmează, pornind de la forma textuală a rezultatelor afișate:

a) `1+1.0`
b) `1/1+1`
c) `9*3 mod 4`
d) `4*x>9*y`
e) `chr(65)`
f) `not(x>y)`
g) `pred(9)>succ(7)`
h) `15 div ord(3)`
i) `trunc(x)+round(6*y)`
j) `sqr(3)-sqrt(16)`

Se consideră că variabilele `x` și `y` sunt de tip `real`.

❹ Se consideră declarațiile:

```
type T1=1..10;
     T2=11..20;
     T3='A'..'Z';
     T4=(A, B, C, D, E, F, G, H);
var  i : T1;
     j : T2;
     k : T3;
     m : 'C'..'G';
     n : T4;
     p : C..G;
     q : boolean;
```

Aflați tipul următoarelor expresii:

a) `i-j`

b) `i div j`

c) `6.3*i`

d) `cos(3*i-6*j)`

e) `4*i>5*j`

f) `k<m`

g) `k<>m`

h) `chr(i)`

i) `ord(k)`

j) `ord(m)`

k) `n>p`

l) `ord(n)`

m) `succ(n)`

n) `pred(p)`

o) `ord(p)`

p) `ord(k)>ord(m)`

q) `(i>j) and q`

r) `not(i+j>0) or q`

3.5. Instrucțiunea de atribuire

Instrucțiunea în studiu are forma:

<*Variabilă*> := <*Expresie*>

Execuția unei instrucțiuni de atribuire presupune:
a) evaluarea expresiei din partea dreaptă;
b) atribuirea valorii obținute variabilei din partea stângă.

Exemple:

1) `x:=1`

2) `y:=x+3`

3) `z:=sin(x)+cos(y)`

4) `p:=not q`

5) `q:=(a<b) or (x<y)`

6) `c:='A'`

De reținut că simbolul ":=" (se citeşte "atribuire") desemnează o atribuire şi nu trebuie confundat cu operatorul de relație "=" (egal).

O atribuire are loc dacă variabila şi rezultatul evaluării expresiei sunt compatibile din punctul de vedere al atribuirii. În caz contrar, se va produce o eroare.

Variabila şi rezultatul evaluării expresiei sunt **compatibile din punctul de vedere al atribuirii**, dacă este adevărată una dintre următoarele afirmații:

1) variabila şi rezultatul evaluării sunt de tipuri identice;

2) tipul rezultatului este un subdomeniu al tipului variabilei;

3) ambele tipuri sunt subdomenii ale aceluiaşi tip, iar rezultatul este în subdomeniul variabilei;

4) variabila este de tip `real`, iar rezultatul – de tip `integer` sau un subdomeniu al acestuia.

Pentru exemplificare, să examinăm următorul program:

```
Program P39;
 { Compatibilitate din punctul de vedere al atribuirii }
type T1=1..10; { subdomeniu de integer }
     T2=5..15; { subdomeniu de integer }
var i : T1;
    j : T2;
    k, m, n : integer;
    x : real;
begin
 write('k=');
 readln(k);
 i:=k; { corect pentru 1<=k<=10 }
 write('m=');
 readln(m);
 j:=m; { corect pentru 5<=m<=15 }
 write('n=');
 readln(n);
 i:=n+5; { corect pentru -4<=n<=5 }
 j:=n+2; { corect pentru 3<=n<=13 }
 x:=i+j;
 writeln('x=', x);
end.
```

Programul va derula fără erori numai pentru următoarele valori de intrare:

$$1 \le k \le 10;\ 5 \le m \le 15;\ 3 \le n \le 5.$$

Evident, în programul `P39` instrucțiunile de tipul

```
k:=x
```

```
i:=x+1
```

```
j:=sin(i)
```

ş.a.m.d. ar fi incorecte, întrucât `x`, `x+1`, `sin(i)` sunt expresii de tip `real`, iar variabilele `k`, `i`, `j` sunt de tip `integer` sau subdomenii ale acestuia.

Întrebări şi exerciţii

❶ Cum se execută o instrucţiune de atribuire?

❷ Explicaţi termenul "compatibilitate din punctul de vedere al atribuirii".

❸ Se consideră următoarele declaraţii:

```
type Zi = (L, Ma, Mi, J, V, S, D);
     Culoare = (Galben, Verde, Albastru, Violet);
var i, j, k : integer;
    z : Zi;
    c : Culoare;
    x : real;
```

Care dintre instrucţiunile ce urmează sunt corecte?

a) `i:=12`

b) `j:=ord(i)`

c) `x:=ord(z)+1`

d) `k:=ord(x)+2`

e) `c:=i+4`

f) `c:=Verde`

g) `z:=D`

h) `c:=Pred(Galben)`

i) `x:=Succ(z)`

j) `i:=Succ(c)`

❹ Precizaţi pentru care valori ale variabilei `j` programul `P40` va derula fără erori?

```
Program P40;
var i : -10..+10;
    j : integer;
begin
 write('j=');
 readln(j);
 i:=j+15;
 writeln('i=', i);
end.
```

3.6. Instrucţiunea *apel de procedură*

Procedura este un subalgoritm scris în limbaj de programare ce poate fi apelat din mai multe puncte ale unui program. Fiecare procedură are un nume, de exemplu, `readln`, `writeln`, `CitireDate`, `A15` ş.a.m.d. Limbajul PASCAL include un set de proceduri predefinite, cunoscute oricărui program: `read`, `readln`, `write`, `writeln`, `get`, `put`, `new` etc. În completare, programatorul poate defini proceduri proprii.

Instrucţiunea *apel de procedură* lansează în execuţie procedura cu numele specificat. Instrucţiunea în studiu are forma:

Nume_procedură (Parametrul_actual_1, Parametrul_actual_2, …)

Dacă procedura nu are parametri, instrucțiunea va conține doar numele procedurii apelate.

Exemple:

1) `readln(x)`

2) `readln(x, y, z)`

3) `writeln(x+y, sin(x))`

4) `CitireDate(f, t)`

5) `Exit`

6) `writeln(2*x)`

Tipul parametrilor actuali şi ordinea în care aceştia apar în listă sunt impuse de declarațiile procedurii respective. În consecință, regulile de formare a listelor de parametri actuali vor fi studiate mai amănunțit în clasele următoare.

Întrebări şi exerciții

❶ Care este destinația instrucțiunii *apel de procedură*?

❷ Se consideră următoarele instrucțiuni:

a) `readln(x, y, z, q)`

b) `CitireDate(ff, tt)`

c) `Halt`

d) `writeln('x=', x, 'y=', y)`

e) `writeln('x+y=', x+y, 'sin(x)=', sin(x))`

Precizați numele procedurilor apelate, numărul de parametri actuali din fiecare apel și parametrii propriu-ziși.

❸ Scrieți un program PASCAL ce citește de la tastatură lungimea laturilor unui triunghi și afișează pe ecran aria acestuia.

3.7. Afișarea informației alfanumerice

În versiunile uzuale ale limbajului PASCAL, ecranul vizualizatorului este desemnat ca dispozitiv-standard de ieşire. De regulă, ecranul este împărțit în zone convenționale, numite zone-caracter. De obicei, aceste zone formează 25 de linii – câte 80 de caractere pe linie. Zona în care va fi afişat caracterul curent este indicată de cursor.

Datele de ieşire ale unui program PASCAL pot fi afişate pe ecran printr-un apel `write(x)` sau `writeln(x)`.

Apelul

```
write(x1, x2, ..., xn)
```

este echivalent cu

```
write(x1); write(x2); ...; write(xn).
```

Parametrii actuali dintr-un apel `write` sau `writeln` se numesc ***parametri de ieşire***. Aceştia pot avea una dintre formele:

e $\qquad$ $e : w$ $\qquad$ $e : w : f$

unde e este o expresie de tip `integer`, `real`, `boolean`, `char` sau *şir de caractere*, a cărei valoare trebuie afişată; w şi f sunt expresii de tip `integer`, numite ***specificatori de format***. Expresia w specifică prin valoarea sa numărul minim de caractere ce vor fi folosite la afişarea valorii lui e; dacă sunt necesare mai puţin de w caractere, atunci forma externă a valorii lui e va fi completată cu spaţii la stânga până la w caractere (*fig. 3.1*).

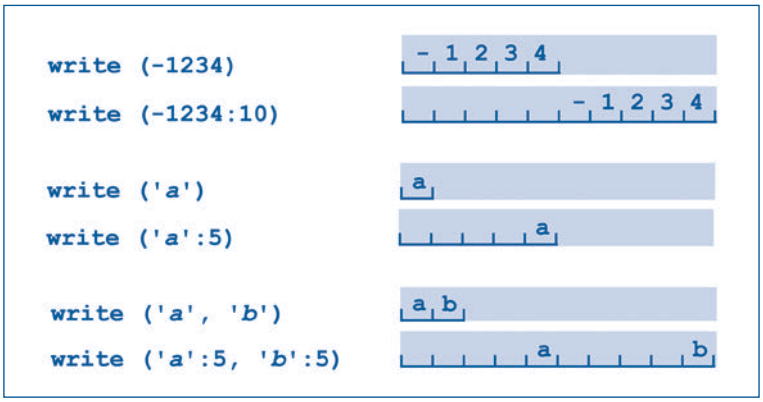


Fig. 3.1. Semnificaţia specificatorului de format w

Specificatorul de format f are sens în cazul în care e este de tip `real` şi indică numărul de cifre care urmează punctul zecimal în scrierea valorii lui e în virgulă fixă, fără factor de scală. În lipsa lui f, valoarea lui e se scrie în virgulă mobilă, cu factor de scală (*fig. 3.2*).

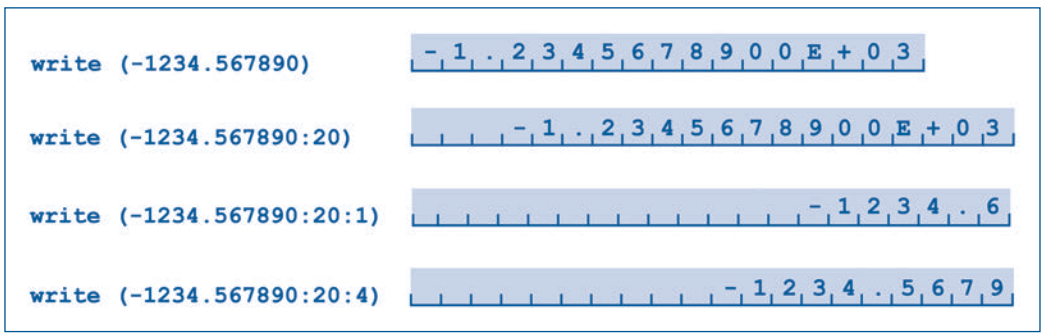


Fig. 3.2. Semnificaţia specificatorului de format f

Diferenţa dintre procedurile `write` şi `writeln` constă în faptul că, după afişarea datelor, `write` lasă cursorul în linia curentă, în timp ce `writeln` îl trece la începutul unei linii noi. Utilizarea raţională a apelurilor `write`, `writeln` şi a specificatorilor de format asigură o afişare lizibilă a datelor de ieşire. Dacă pe ecran se afişează mai multe valori, se recomandă ca acestea să fie însoţite de identificatorii respectivi sau de mesaje sugestive.

Exemple:

1) `write('Suma numerelor introduse este')`

2) `writeln(s:20)`

3) `writeln('Suma=', s)`

4) `writeln('s=', s)`

5) `writeln('x=', x, 'y=':5, y, 'z=':5, z)`

Întrebări şi exerciţii

❶ Care este destinaţia specificatorului de format?

❷ Cum se numesc parametrii actuali ai unui apel de procedură `write` sau `writeln`?

❸ Precizaţi formatul datelor afişate pe ecran de programele ce urmează:

```
Program P41;
 {Afisarea datelor de tip integer }
var i : integer;
begin
 i:=-1234;
 writeln(i);
 writeln(i:1);
 writeln(i:8);
 writeln(i, i);
 writeln(i:8, i:8);
 writeln(i, i, i);
 writeln(i:8, i:8, i:8);
end.
```

```
Program P42;
 {Afisarea datelor de tip real }
var x : real;
begin
 x:=-1234.567890;
 writeln(x);
 writeln(x:20);
 writeln(x:20:1);
 writeln(x:20:2);
 writeln(x:20:4);
 writeln(x, x, x);
 writeln(x:20, x:20, x:20);
 writeln(x:20:4, x:20:4, x:20:4);
end.
```

```
Program P43;
 { Afisarea datelor de tip boolean }
var p : boolean;
begin
 p:=false;
 writeln(p);
 writeln(p:10);
 writeln(p, p);
 writeln(p:10, p:10);
end.
```

```
Program P44;
 { Afisarea sirurilor de caractere }
begin
 writeln('abc');
 writeln('abc':10);
 writeln('abc', 'abc');
 writeln('abc':10, 'abc':10);
end.
```

❹ Elaboraţi un program care afişează pe ecran valorile `1234567890`, `123`, `123.0` şi `true` după cum urmează:

```
1234567890
123
123.0
true
1234567890
                  123
                  123.000
                  true
```

3.8. Citirea datelor de la tastatură

În mod obişnuit, tastatura vizualizatorului este desemnată ca **dispozitiv-standard de intrare**. Citirea datelor de la tastatură se realizează prin apelul procedurilor predefinite `read` sau `readln`. Lista parametrilor actuali ai unui apel `read` sau `readln` poate să includă variabile de tip `integer`, `real`, `char`, inclusiv *şir de caractere*.

Apelul

```
read(x)
```

are următorul efect. Dacă variabila `x` este de tip `integer` sau `real`, atunci este citit întregul şir de caractere care reprezintă valoarea întreagă sau reală. Dacă `x` este de tip `char`, procedura citeşte un singur caracter.

Apelul

```
read(x1, x2, ..., xn)
```

este echivalent cu

```
read(x1); read(x2); ...; read(xn).
```

Datele numerice introduse de la tastatură trebuie separate prin spații sau caractere de sfârşit de linie. Spațiile dinaintea unei valori numerice sunt ignorate. Şirul de caractere care reprezintă o valoare numerică se conformează sintaxei constantelor numerice de tipul respectiv. În caz contrar, este semnalată o eroare de intrare-ieşire.

De exemplu, fie programul:

```
Program P45;
 { Citirea datelor numerice de la tastatura }
var i, j : integer;
    x, y : real;
begin
  read(i, j, x, y);
  writeln('Ati introdus:');
  writeln('i=', i);
  writeln('j=', j);
  writeln('x=', x);
  writeln('y=', y);
end.
```

în care sunt citite de la tastatură valorile variabilelor `i`, `j`, `x`, `y`. După lansarea programului în execuție, utilizatorul tastează:

```
1<ENTER>
2<ENTER>
3.0<ENTER>
4.0<ENTER>
```

Pe ecran se va afişa:

```
Ati introdus:
i=1
j=2
x=3.0000000000E+00
y=4.0000000000E+00
```

Acelaşi efect se va obține şi la tastarea numerelor într-o singură linie:

```
1 2 3.0 4.0<ENTER>
```

Dacă e necesar, numerele întregi, introduse de utilizator, sunt convertite în valori reale. De exemplu, în cazul programului `P45` utilizatorul poate tasta

```
1 2 3 4<ENTER>
```

Procedura `readln` citeşte datele în acelaşi mod ca şi procedura `read`. Însă după citirea ultimei valori, restul caracterelor din linia curentă se ignoră. Pentru exemplificare, prezentăm programul `P46`:

```
Program P46;
 { Apelul procedurii readln }
var i, j : integer;
    x, y : real;
begin
  writeln('Apelul procedurii read');
  read(i, j);
  read(x, y);
  writeln('Ati introdus:');
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);
  writeln('Apelul procedurii readln');
  readln(i, j);
  readln(x, y);
  writeln('Ati introdus:');
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);
end.
```

La execuţia instrucţiunilor

```
read(i, j);
read(x, y);
```

valorile numerice din linia introdusă de utilizator

```
1 2 3 4<ENTER>
```

vor fi atribuite variabilelor, respectiv, `i`, `j`, `x`, `y`. La execuţia instrucţiunii

```
readln(i, j)
```

valorile numerice 1 şi 2 din linia

```
1 2 3 4<ENTER>
```

vor fi atribuite variabilelor `i` şi `j`. Numerele `3` şi `4` se ignoră. În continuare calculatorul execută instrucţiunea

```
readln(x, y)
```

adică va aştepta introducerea unor valori pentru `x` şi `y`.

Subliniem faptul că apelul procedurii `readln` fără parametri va forţa calculatorul să aştepte acţionarea tastei `<ENTER>`. Acest apel se foloseşte pentru a suspenda derularea programului, oferindu-i utilizatorului posibilitatea să analizeze rezultatele afişate anterior pe ecran.

Pentru a înlesni introducerea datelor, se recomandă ca apelurile `read` (...) şi `readln` (...) să fie precedate de afişarea unor mesaje sugestive.

Exemple:

1) `write('Dati doua numere:'); readln(x, y);`

2) `write('Dati un numar intreg:'); readln(i);`

3) `write('x='); readln(x);`

4) `write('Raspundeti D sau N:'); readln(c);`

Întrebări şi exerciţii

❶ Cum se separă datele numerice care se introduc de la tastatură?

❷ Care este diferenţa dintre procedurile `read` şi `readln`?

❸ Se consideră următorul program:

```
Program P47;
var i : integer;
    c : char;
    x : real;
begin
 readln(i);
 readln(c);
 readln(x);
 writeln('i=', i);
 writeln('c=', c);
 writeln('x=', x);
 readln;
end.
```

Precizaţi rezultatele afişate de acest program după tastarea datelor de intrare:

a)
```
1
2
3
```

b)
```
1 2 3
5 6 7
8 9 0
```

c)
```
123
456
789
```

d)
```
123 456 789
abc def ghi
890 abc def
```

3.9. Instrucţiunea de efect nul

Executarea acestei instrucţiuni nu are niciun efect asupra variabilelor programului.

În textul unui program instrucţiunea de efect nul nu este reprezentată prin nimic. Întrucât instrucţiunile unui program sunt despărţite între ele prin delimitatorul ";", prezenţa instrucţiunii de efect nul este marcată de apariţia acestui delimitator.

De exemplu, în textul

```
x:=4;;;; y:=x+1
```

există 5 instrucţiuni, dintre care 3 de efect nul.

În mod obişnuit, instrucţiunea de efect nul se utilizează la etapa elaborării şi depanării unor programe complexe. Deşi efectul său la execuţie este nul, inserarea sau eliminarea unei astfel de instrucţiuni (mai exact, a simbolului ";") poate să altereze semnificaţia programului.

3.10. Instrucţiunea `if`

Instrucțiunea de ramificare simplă **`if`**, în funcție de valoarea unei expresii de tip `boolean`, decide fluxul execuției.

Forma simplă a acestei instrucțiuni este:

`if` *Expresie booleană* **`then`** *Instrucțiune*

iar forma completă include şi alternativa **`else`**:

`if` *Expresie booleană* **`then`** *Instrucțiunea_1* **`else`** *Instrucțiunea_2*

Expresia booleană din componența instrucțiunii **`if`** se numeşte **condiție**.

Execuția instrucțiunii **`if`** începe prin evaluarea condiției. Dacă rezultatul evaluării este `true`, atunci se execută instrucțiunea situată după cuvântul-cheie **`then`**. Dacă condiția are valoarea `false`, atunci: fie că se execută instrucțiunea situată după cuvântul-cheie **`else`** (dacă există), fie că se trece la instrucțiunea situată după instrucțiunea **`if`**.

În programul ce urmează, instrucțiunea **`if`** este utilizată pentru determinarea maximului a două numere `x` şi `y`, citite de la tastatură.

```
Program P48;
{ Determinarea maximului a doua numere }
var x, y, max : real;
begin
 writeln('Dati doua numere:');
 write('x='); readln(x);
 write('y='); readln(y);
 if x>=y then max:=x else max:=y;
 writeln('max=', max);
 readln;
end.
```

Următorul program transformă cifrele romane *I* (unu), *V* (cinci), *X* (zece), *L* (cincizeci), *C* (o sută), *D* (cinci sute) sau *M* (o mie), citite de la tastatură, în numerele corespunzătoare din sistemul zecimal.

```
Program P49;
 { Conversia cifrelor romane }
var i : integer; c : char;
begin
 i:=0;
 writeln('Introduceti una dintre cifrele');
 writeln('romane I, V, X, L, C, D, M');
 readln(c);
 if c='I' then i:=1;
 if c='V' then i:=5;
```

```
 if c='X' then i:=10;
 if c='L' then i:=50;
 if c='C' then i:=100;
 if c='D' then i:=500;
 if c='M' then i:=1000;
 if i=0 then writeln(c, ' - nu este o cifra romana')
        else writeln(i);
 readln;
end.
```

De reținut că limbajul PASCAL nu consideră simbolul ";" ca făcând parte din instrucțiune, ci îl folosește ca delimitator. Prin urmare, dacă într-o instrucțiune

```
if B then S
```

introducem înainte de S instrucțiunea cu efect nul

```
if B then; S
```

atunci S nu mai intră în componența instrucțiunii condiționale, deci este executată indiferent de valoarea lui B.

Dacă într-o instrucțiune

```
if B then I else J
```

introducem după I simbolul ";", obținem un program incorect sub aspect sintactic:

```
if B then I; else J
```

În acest caz, secvența **else J** este interpretată ca fiind instrucțiunea ce urmează celei condiționale.

Întrebări și exerciții

❶ Care este destinația instrucțiunii **if**?

❷ Ce valori va lua variabila x după executarea fiecăreia dintre instrucțiunile ce urmează? Se consideră că a=18, b=-15 și p=true.

a)
```
if a>b then x:=1 else x:=4;
```

b)
```
if a<b then x:=15 else x:=-21;
```

c)
```
if p then x:=32 else x:=638;
```

d)
```
if not p then x:=0 else x:=1;
```

e)
```
if (a<b) and p then x:=-1 else x:=1;
```

f)
```
if (a>b) or p then x:=-6 else x:=-5;
```

g)
```
if not (a>b) then x:=19 else x:=-2;
```

h)
```
if (a=b) or p then x:=89 else x:=-15.
```

❸ Elaborați un program care calculează valorile uneia dintre funcțiile:

a) $y=\begin{cases}2x, x\geq 0;\\ \dfrac{x}{2}, x<0;\end{cases}$ *b)* $y=\begin{cases}x+3, x>5;\\ x-3, x\leq 5;\end{cases}$

c) $y=\begin{cases}x, x\geq 3;\\ x+4, x<3;\end{cases}$ *d)* $y=\begin{cases}x, |x|>5;\\ 2x, |x|\leq 5.\end{cases}$

Exemplu: $y=\begin{cases}x+6, x>4;\\ x-3, x\leq 4.\end{cases}$

```
Program P50;
var x, y : real;
begin
 write('x='); readln(x);
 if x>4 then y:=x+6 else y:=x-3;
 writeln('y=', y);
 readln;
end.
```

❹ Ce rezultate va afișa următorul program?

```
Program P51;
var x, y : real;
begin
 write('x='); readln(x);
 y:=x;
 if x>0 then; y:=2*x;
 writeln('y=', y);
 readln;
end.
```

❺ Comentați mesajele afișate pe ecran pe parcursul compilării programului `P52`:

```
Program P52;
 { Eroare }
var x, y : real;
begin
 write('x=');
 readln(x);
 if x>4 then y:=x+6;
        else y:=x-3;
 writeln('y=', y); readln;
end.
```

❻ Scrieți un program care transformă numerele zecimale 1, 5, 10, 50, 100, 500 și 1000, citite de la tastatură, în cifre romane.

3.11. Instrucţiunea case

Instrucțiunea de ramificare multiplă **case** conține o expresie numită **selector** şi o listă de instrucțiuni. Fiecare instrucțiune este prefixată de una sau mai multe constante de caz. Instrucțiunea în studiu are forma:

case *Expresie* **of**
    Constanta_1: Instrucțiune ;
    Constanta_2: Instrucțiune ;
    …
end

Selectorul trebuie să fie de tip ordinal. Toate constantele de caz trebuie să fie unice şi compatibile, din punctul de vedere al atribuirii, cu tipul selectorului.

Exemple:

```
var i : integer; c : char; a, b, y : real;
```

1)
```
case i of
  0, 2, 4, 6, 8 : writeln('Cifra para');
  1, 3, 5, 7, 9 : writeln('Cifra impara');
end
```

2)
```
case c of
  '+' : y:=a+b;
  '-' : y:=a-b;
  '*' : y:=a*b;
  '/' : y:=a/b;
end
```

Execuția instrucțiunii **case** începe prin evaluarea selectorului. În funcție de valoarea obținută, se execută instrucțiunea prefixată de constanta respectivă.

În programul ce urmează, instrucțiunea **case** este utilizată pentru conversia cifrelor romane în numere zecimale.

```
Program P53;
 { Conversia cifrelor romane }
var i : integer; c : char;
begin
 i:=0;
 writeln('Introduceti una din cifrele');
 writeln('romane I, V, X, L, C, D, M');
 readln(c);
 case c of
   'I' : i:=1;
   'V' : i:=5;
   'X' : i:=10;
   'L' : i:=50;
   'C' : i:=100;
```

```
    'D' : i:=500;
    'M' : i:=1000;
  end;
  if i=0 then writeln(c, ' - nu este o cifra romana')
         else writeln(i);
  readln;
end.
```

Subliniem faptul că în unele implementări ale limbajului sintaxa și semantica instrucțiunii **case** au fost modificate. Lista cazurilor poate să includă o instrucțiune precedată de cuvântul-cheie **else** (în unele versiuni **otherwise**). Constantele de caz pot fi înlocuite cu intervale de forma:

<*Constantă*>..<*Constantă*>

Exemplu (Turbo PASCAL 7.0, Free PASCAL):

```
Program P54;
 { Simularea unui calculator de buzunar }
var a, b : real;
    c : char;
begin
 write('a='); readln(a);
 write('b='); readln(b);
 write('Cod operatie '); readln(c);
 case c of
   '+' : writeln('a+b=', a+b);
   '-' : writeln('a-b=', a-b);
   '*' : writeln('a*b=', a*b);
   '/' : writeln('a/b=', a/b);
 else writeln('Cod operatie necunoscut');
 end;
 readln;
end.
```

Întrebări și exerciții

❶ Cum se execută o instrucțiune **case**? De ce tip trebuie să fie selectorul?

❷ Ce fel de constante pot fi utilizate în calitate de constante de caz?

❸ Înlocuiți instrucțiunea **case** a programului P54 cu o secvență echivalentă de instrucțiuni **if**.

❹ Utilizând instrucțiunea **case**, scrieți un program care transformă numerele zecimale 1, 5, 10, 50, 100, 500, 1000, citite de la tastatură, în cifre romane.

❺ Ce va apare pe ecran în urma execuției programului P55?

```
  Program P55;
  type Semnal=(Rosu, Galben, Verde);
  var s : Semnal;
```

```
begin
 s:=Verde;
 s:=pred(s);
 case s of
   Rosu : writeln('STOP');
   Galben : writeln('ATENTIE');
   Verde : writeln('START');
 end;
 readln;
end.
```

❻ Comentați programele:

```
Program P56;
 { Eroare }
var x : real;
begin
 writeln('x='); readln(x);
 case x of
   0,2,4,6,8 : writeln('Cifra para');
   1,3,5,7,9 : writeln('Cifra impara');
 end;
 readln;
end.
```

```
Program P57;
 { Eroare }
var i : 1..4;
begin
 write('i='); readln(i);
 case i of
   1 : writeln('unu');
   2 : writeln('doi');
   3 : writeln('trei');
   4 : writeln('patru');
   5 : writeln('cinci');
 end;
 readln;
end.
```

```
Program P58;
 { Eroare }
type Semnal = (Rosu, Galben, Verde);
     Culoare = (Albastru, Portocaliu);
var s : Semnal;
    c : Culoare;
begin
 { ... }
```

```
  case s of
    Rosu : writeln('STOP');
    Galben : writeln('ATENTIE');
    Verde : writeln('START');
    Albastru : writeln('PAUZA');
  end;
  { ... }
end.
```

3.12. Instrucţiunea for

Instrucțiunea **for** indică execuția repetată a unei instrucțiuni în funcție de valoarea unei variabile de control. Această instrucțiune are două forme:

for *Variabilă* := *Expresia_1* **to** *Expresia_2* **do** *Instrucțiune*

for *Variabilă* := *Expresia_1* **downto** *Expresia_2* **do** *Instrucțiune*

Variabila situată după cuvântul-cheie **for** se numeşte **variabilă de control** sau **contor**. Această variabilă trebuie să fie de tip ordinal.

Valorile expresiilor din componența instrucțiunii **for** trebuie să fie compatibile, în aspectul atribuirii, cu tipul variabilei de control. Aceste expresii sunt evaluate o singură dată, la începutul ciclului. Prima expresie indică valoarea inițială, iar expresia a doua – valoarea finală a variabilei de control.

Instrucțiunea situată după cuvântul-cheie **do** se execută pentru fiecare valoare din domeniul determinat de valoarea inițială şi de valoarea finală.

Dacă instrucțiunea **for** utilizează pasul **to**, valorile variabilei de control sunt incrementate la fiecare repetiție, adică se trece la succesorul valorii curente. Dacă valoarea inițială este mai mare decât valoarea finală, instrucțiunea situată după cuvântul-cheie **do** nu se execută niciodată.

Dacă instrucțiunea **for** utilizează pasul **downto**, valorile variabilei de control sunt decrementate la fiecare repetiție, adică se trece la predecesorul valorii curente. Dacă valoarea inițială este mai mică decât valoarea finală, instrucțiunea situată după cuvântul-cheie **do** nu se execută niciodată.

Exemplu:

```
Program P59;
 { Instructiunea for }
var i : integer;
    c : char;
begin
  for i:=0 to 9 do write(i:2);
  writeln;
  for i:=9 downto 0 do write(i:2);
  writeln;
  for c:='A' to 'Z' do write(c:2);
```

```
  writeln;
  for c:='Z' downto 'A' do write(c:2);
  writeln;
  readln;
end.
```

Rezultatele afişate pe ecran:

```
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
```

Valorile variabilei de control nu pot fi modificate în interiorul ciclului, adică:

1) nu se fac atribuiri variabilei de control;

2) variabila actuală de control nu poate fi variabilă de control a altei instrucțiuni **for** incluse;

3) nu se admit apeluri de tipul `read`, `readln` în care apare variabila de control.

La ieşirea din instrucțiunea **for**, valoarea variabilei de control nu este definită, în afara cazului când ieşirea din ciclu se face forțat, printr-o instrucțiune de salt necondiționat **goto**.

Instrucțiunea **for** este utilă pentru programarea algoritmilor iterativi, în care numărul de repetări este cunoscut. Pentru exemplificare prezentăm programele `P60`, `P61` şi `P62` care calculează, respectiv, $n!$, x^n şi suma $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}$.

```
Program P60;
 { Calcularea factorialului }
var n, i, f : 0..MaxInt;
begin  write('n='); readln(n);
  f:=1;
  for i:=1 to n do f:=f*i;
  writeln('n!=', f);
  readln;
end.
```

```
Program P61;
 { Calcularea lui x la puterea n }
var x, y : real;
    n, i : 0..MaxInt;
begin
  write('x='); readln(x);
  write('n='); readln(n);
  y:=1;
  for i:=1 to n do y:=y*x;
  writeln('y=', y);
  readln;
end.
```

```
Program P62;
 { Calcularea sumei 1 + 1/2 + 1/3 + ... + 1/n }
var n, i : 1..MaxInt;
    s : real;
begin
  write('n=');
  readln(n);
  s:=0;
  for i:=1 to n do s:=s+1/i;
  writeln('s=', s);
  readln;
end.
```

Întrebări şi exerciţii

❶ Cum se execută o instrucțiune **for**?

❷ Ce va afișa pe ecran programul P63?

```
Program P63;
type Zi = (L, Ma, Mi, J, V, S, D);
var z : Zi;
begin
  for z:=L to S do writeln(ord(z));
  readln;
  for z:=D downto Ma do writeln(ord(z));
  readln;
end.
```

❸ Se consideră declarațiile:

```
var i, j, n : integer;
    x, y : real;
    c : char;
```

Care dintre instrucțiunile ce urmează sunt corecte?

a) `for i:=-5 to 5 do j:=i+3;`

b) `for i:=-5 to 5 do i:=j+3;`

c) `for j:=-5 to 5 do i:=j+3;`

d) `for i:=1 to n do y:=y/i;`

e) `for x:=1 to n do y:=y/x;`

f) `for c:='A' to 'Z' do writeln(ord(c));`

g) `for c:='Z' downto 'A' do writeln(ord(c));`

h)
```
for i:=-5 downto -10 do readln(i);
```

i)
```
for i:=ord('A') to ord('A')+ 9 do writeln(i);
```

j)
```
for c:='0' to '9' do writeln(c, ord(c):3);
```

k)
```
for j:=i/2 to i/2+10 do writeln(j).
```

❹ Se consideră declarațiile:

```
var i, m, n : integer;
```

De câte ori se vor executa apelurile `writeln(i)` și `writeln(2*i)` din componența instrucțiunilor

```
for i:=m to n do writeln(i);
for i:=m to n do writeln(2*i);
```

dacă:

a) `m=1, n=5;`

b) `m=3, n=5;`

c) `m=3, n=3;`

d) `m=5, n=3?`

❺ Elaborați un program care afișează pe ecran codurile caracterelor *'A'*, *'B'*, *'C'*, ..., *'Z'*.

❻ Calculați pentru primii *n* termeni:

a) $1 + 3 + 5 + 7 + \ldots$ și $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$;

b) $2 + 4 + 6 + 8 + \ldots$ și $2 \cdot 4 \cdot 6 \cdot 8 \cdot \ldots$;

c) $3 + 6 + 9 + 12 + \ldots$ și $3 \cdot 6 \cdot 9 \cdot 12 \cdot \ldots$;

d) $4 + 8 + 12 + 16 + \ldots$ și $4 \cdot 8 \cdot 12 \cdot 16 \cdot \ldots$.

Exemplu: Pentru `n=3` avem $1 + 3 + 5 = 9$; $1 \cdot 3 \cdot 5 = 15$.

❼ Calculați suma primilor *n* termeni:

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\frac{1}{6}+\ldots.$$

Indicație: Utilizați în ciclu instrucțiunea **if** `odd(...)` **then** ... **else** ... .

3.13. Instrucţiunea compusă

Instrucțiunea în studiu are forma:

begin
Instrucțiunea_1;
Instrucțiunea_2;
…
end

Exemple:

1)
```
begin
  a:=x+12;
  p:=q and r;
  writeln(p)
end
```

2)
```
begin
  write('x=');
  readln(x)
end
```

Cuvintele-cheie **begin** şi **end** joacă rolul unor "paranteze". Mulţimea instrucţiunilor cuprinse între aceste paranteze, din punctul de vedere al limbajului, formează o singură instrucţiune. Prin urmare, instrucţiunea compusă este utilă pentru a plasa mai multe instrucţiuni în locurile din programe în care este permisă numai o singură instrucţiune (vezi instrucţiunile **if**, **case**, **for** etc.).

Exemple:

1)
```
if a>0 then begin x:=a+b; y:=a*b end
       else begin x:=a-b; y:=a/b end;
```

2)
```
case c of
  '+' : begin y:=a+b; writeln('Adunarea') end;
  '-' : begin y:=a-b; writeln('Scaderea') end;
  '*' : begin y:=a*b; writeln('Inmultirea') end;
  '/' : begin y:=a/b; writeln('Impartirea') end;
end;
```

3)
```
for i:=1 to n do
  begin
    write('x=');
    readln(x);
    s:=s+x
  end;
```

De menţionat că partea executabilă a oricărui program este o instrucţiune compusă, adică o secvenţă de instrucţiuni încadrată între "parantezele" **begin** şi **end**.

Întrucât simbolul "**;**" nu termină, ci separă instrucţiunile, el nu este necesar înaintea cuvântului-cheie **end**. Cu toate acestea, mulţi programatori inserează acest simbol cu scopul de a continua, în caz de necesitate, lista respectivă de instrucţiuni. Amintim că apariţia adiţională a unui simbol ";" semnifică inserarea unei instrucţiuni de efect nul.

Pentru a face programele mai lizibile, cuvintele **begin** şi **end** se scriu strict unul sub altul, iar instrucţiunile dintre "paranteze" – cu câteva poziţii mai la dreapta. Dacă instrucţiunea compusă **begin...end** este inclusă în componenţa altor instrucţiuni (**if**, **case**, **for** etc.), cuvintele-cheie **begin** şi **end** se scriu deplasate la dreapta.

Pentru exemplificare, prezentăm programul `P64`, în care se calculează media aritmetică a *n* numere, citite de la tastatură.

```
Program P64;
 { Media aritmetica a n numere }
var x, suma, media : real;
    i, n : integer;
begin
 write('n='); readln(n);
 suma:=0;
 writeln('Dati ', n, ' numere:');
 for i:=1 to n do
   begin
    write('x='); readln(x);
    suma:=suma+x;
   end;
 if n>0 then
          begin
            media:=suma/n;
            writeln('media=', media);
          end
        else writeln('media=*****');
 readln;
end.
```

Întrebări şi exerciţii

❶ Care este destinaţia instrucţiunii compuse?

❷ Elaboraţi un program care citeşte de la tastatură *n* numere şi afişează pe ecran:
a) suma şi media aritmetică a numerelor citite;
b) suma şi media aritmetică a numerelor pozitive;
c) suma şi media aritmetică a numerelor negative.

❸ Elaboraţi un program care citeşte de la tastatură *n* caractere şi afişează pe ecran:
a) numărul cifrelor zecimale citite;
b) numărul cifrelor pare;
c) numărul cifrelor impare;
d) numărul literelor citite;
e) numărul vocalelor;
f) numărul consoanelor.

Caracterele introduse se separă prin acţionarea tastei <ENTER>. Sunt admise cifrele zecimale 0, 1, 2, ..., 9 şi literele mari *A*, *B*, *C*, ..., *Z* ale alfabetului latin.

3.14. Instrucţiunea `while`

Instrucţiunea **`while`** conţine o expresie booleană care controlează execuţia repetată a altei instrucţiuni. Forma instrucţiunii în studiu este:

`while` *Expresie booleană* **`do`** *Instrucţiune*

Exemple:

1)
```
while x>0 do x:=x-1;
```

2)
```
while x<3.14 do
  begin
    x:=x+0.001;
    writeln(sin(x));
  end;
```

3)
```
while p do
  begin
    x:=x+0.001;
    y:=10*x;
    p:=y<1000;
  end;
```

Instrucțiunea situată după cuvântul-cheie **do** se execută repetat atâta timp, cât valoarea expresiei booleene este `true`. Dacă expresia booleană ia valoarea `false`, instrucțiunea de după **do** nu se mai execută. Se recomandă ca expresia booleană să fie cât mai simplă, deoarece ea este evaluată la fiecare iterație.

În mod obișnuit, instrucțiunea **while** se folosește pentru organizarea calculelor repetitive cu variabile de control de tip `real`.

În programul ce urmează, instrucțiunea **while** este utilizată pentru afișarea valorilor funcției $y = 2x$. Argumentul x ia valori de la x_1 la x_2 cu pasul Δx.

```
Program P65;
 { Tabelul functiei y=2*x }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      y:=2*x;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
 readln;
end.
```

Instrucțiunea **while** se consideră deosebit de utilă în situația în care numărul de execuții repetate ale unei secvențe de instrucțiuni este dificil de evaluat.

Pentru exemplificare, prezentăm programul `P66`, care afișează pe ecran media aritmetică a numerelor pozitive citite de la tastatură.

```
Program P66;
 { Media aritmetica a numerelor pozitive
   citite de la tastatura }
var x, suma : real;
    n : integer;
begin
  n:=0;
  suma:=0;
  writeln('Dati numere pozitive:');
  readln(x);
  while x>0 do
    begin
      n:=n+1;
      suma:=suma+x;
      readln(x);
    end;
  writeln('Ati introdus ', n, ' numere pozitive.');
  if n>0 then writeln('media=', suma/n)
         else writeln('media=*****');
  readln;
end.
```

Se observă că numărul de execuții repetate ale instrucțiunii compuse **begin ... end** din componența instrucțiunii **while** nu poate fi calculat din timp. Execuția instrucțiunii **while** se termină când utilizatorul introduce un număr $x \leq 0$.

Întrebări și exerciții

❶ Cum se execută o instrucțiune **while**?

❷ Utilizând instrucțiunea **while**, scrieți un program care afișează pe ecran valorile funcției y pentru valori ale argumentului de la x_1 la x_2 cu pasul Δx:

a) $y = \frac{x}{3} + 2;$ *b)* $y = \frac{x}{2};$

c) $y = 3x - 4;$ *d)* $y = 4x - 13.$

❸ Utilizatorul introduce de la tastatură numere întregi pozitive, separate prin acționarea tastei <ENTER>. Sfârșitul secvenței de numere e indicat prin introducerea numărului 0. Scrieți un program care afișează pe ecran:
a) suma și media aritmetică a numerelor pare;
b) suma și media aritmetică a numerelor impare.

❹ Scrieți un program care afișează pe ecran valorile funcției $y = f(x)$. Argumentul x ia valori de la x_1 la x_2 cu pasul Δx:

a) $y = \begin{cases} x, & x > 3; \\ 2x, & x \leq 3; \end{cases}$ *b)* $y = \begin{cases} 6x, & x \geq 0; \\ 4x, & x < 0; \end{cases}$

c) $y=\begin{cases} x+6, & x>5; \\ x-6, & x\le 5; \end{cases}$ *d)* $y=\begin{cases} 3-x, & x>4; \\ 3+x, & x\le 4. \end{cases}$

Exemplu: $y=\begin{cases} x+1, & x>8; \\ x-2, & x\le 8. \end{cases}$

```
Program P67;
{ Tabelul functiei }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      if x>8 then y:=x+1 else y:=x-2;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
  readln;
end.
```

❺ Instrucțiunea repetitivă

```
for i:=i1 to i2 do writeln(ord(i))
```

este echivalentă cu secvența de instrucțiuni

```
i:=i1;
while i<=i2 do
  begin
    writeln(ord(i));
    i:=succ(i);
  end.
```

Scrieți o secvență echivalentă pentru instrucțiunea repetitivă

```
for i:=i1 downto i2 do writeln(ord(i))
```

❻ Se consideră declarațiile:

```
var x1, x2, deltaX : real;
    i, n : integer;
```

Care dintre secvențele de instrucțiuni ce urmează sunt echivalente?

a)

```
x:=x1;
while x<=x2 do
begin
  writeln(x);
  x:=x+deltaX;
end;
```

b)
```
n:=trunc((x2-x1)/deltaX)+1;
x:=x1;
for i:=1 to n do
  begin
    writeln(x);
    x:=x+deltaX;
  end;
```

c)
```
n:=round((x2-x1)/deltaX)+1;
x:=x1;
  for i:=1 to n do
    begin
      writeln(x);
      x:=x+deltaX;
    end;
```

Argumentați răspunsul.

3.15. Instrucţiunea `repeat`

Instrucțiunea **`repeat`** indică repetarea unei secvențe de instrucțiuni în funcție de valoarea unei expresii booleene. Forma acestei instrucțiuni este:

`repeat`
Instrucțiunea_1;
Instrucțiunea_2;
...
`until` <*Expresie booleană*>

Exemple:

1)
```
repeat x:=x-1 until x<0;
```

2)
```
repeat
  y:=y+delta;
  writeln(y)
until y>20.5;
```

3)
```
repeat
  readln(i);
  writeln(odd(i))
until i=0;
```

Instrucțiunile situate între cuvintele-cheie **`repeat`** şi **`until`** se execută repetat atâta timp cât expresia booleană este falsă. Când această expresie devine adevărată, se trece la instrucțiunea următoare. Instrucțiunile aflate între **`repeat`** şi **`until`** vor fi executate cel puțin o dată, deoarece evaluarea expresiei logice are loc după ce s-a executat această secvență.

În mod obişnuit, instrucțiunea **`repeat`** se utilizează în locul instrucțiunii **`while`** atunci când evaluarea expresiei care controlează repetiția se face după executarea secvenței de repetat.

Programul ce urmează afişează pe ecran paritatea numerelor întregi citite de la tastatură.

```
Program P68;
 { Paritatea numerelor citite de la tastatura }
var i : integer;
begin
 writeln('Dati numere intregi:');
 repeat
   readln(i);
   if odd(i) then writeln(i:6, ' - numar impar')
             else writeln(i:6, ' - numar par');
 until i=0;
 readln;
end.
```

Execuția instrucțiunii **repeat** se termină când utilizatorul introduce `i=0`.

Instrucțiunea **repeat** este utilizată foarte des pentru validarea (verificarea corectitudinii) datelor introduse de la tastatură.

De exemplu, presupunem că se cere scrierea unui program care citeşte de la tastatură numărul real x şi afişează pe ecran rădăcina pătrată $y = \sqrt{x}$. Evident, valorile negative ale variabilei x sunt inadmisibile.

```
Program P69;
 { Calcularea radacinii patrate }
var x, y : real;
begin
  repeat
    write('Introduceti numarul nenegativ  x=');
    readln(x);
  until x>=0;
  y:=sqrt(x);
  writeln('Radacina patrata  y=', y);
  readln;
end.
```

După lansarea programului `P69` în execuție, utilizatorul este invitat să introducă un număr mai mare sau egal cu zero. Dacă, din greşeală, utilizatorul va introduce un număr negativ, instrucțiunile `write` şi `readln` din componența instrucțiunii **repeat** vor fi executate din nou. Procesul repetitiv va continua până când utilizatorul nu va introduce un număr corect.

Din exemplele studiate se observă că instrucțiunea **repeat** este utilă în situația în care numărul de executări repetate ale unei secvențe de instrucțiuni este dificil de evaluat.

Întrebări şi exerciții

❶ Cum se execută o instrucțiune **repeat**?

❷ Se consideră instrucțiunile:

a)
```
repeat
   <Instructiune 1>;
   <Instructiune 2>;
   ...
   <Instructiune n>;
until p
```

b)
```
while not p do
begin
   <Instructiune 1>;
   <Instructiune 2>;
   ...
   <Instructiune n>;
end.
```

Sunt oare echivalente aceste instrucțiuni? Argumentați răspunsul.

❸ Elaborați un program care citește de la tastatură o secvență de caractere și afișează pe ecran:
a) numărul cifrelor zecimale citite;
b) numărul cifrelor pare;
c) numărul cifrelor impare.
Caracterele introduse se separă prin acționarea tastei <ENTER>. Sunt admise cifrele zecimale 0, 1, 2, ..., 9 și caracterul * care indică sfârșitul secvenței.

❹ Elaborați un program care citește de la tastatură numărul real x și afișează pe ecran valoarea expresiei $\frac{1}{x}$. Evident, valoarea $x = 0$ este inadmisibilă. Validați datele introduse de la tastatură cu ajutorul instrucțiunii **repeat**.

❺ Este oare echivalentă instrucțiunea

```
for i:=i1 to i2 do writeln(ord(i))
```

cu secvența de instrucțiuni ce urmează?

```
i:=i1;
repeat
  writeln(ord(i));
  i:=succ(i);
until i>i2;
```

Argumentați răspunsul.

❻ Scrieți un program care afișează pe ecran valorile funcției $y = f(x)$. Argumentul x ia valori de la x_1 la x_2 cu pasul Δx. Ciclul se va organiza cu ajutorul instrucțiunii **repeat**.

a) $y = 2x;$

b) $y = \frac{x}{3} + 9;$

c) $y = x - 4;$

d) $y = \frac{x}{8} - 6.$

❼ Creați un program care citește de pe tastatură o secvență de caractere și afișează pe ecran:
a) numărul literelor citite;
b) numărul literelor mari;
c) numărul literelor mici.
Caracterele introduse se separă prin acționarea tastei <ENTER>. Sunt admise literele mari și mici ale alfabetului latin și caracterul * care indică sfârșitul secvenței.

Capitolul 4

EDITAREA IMAGINILOR

Studierea materiilor din acest capitol se va baza, în principal, pe metoda învățării prin acțiune. Ce înseamnă acest lucru? Fiecare din voi va trebui să-şi dezvolte, lucrând în mod individual sau în echipă, competența de a învăța să înveți. Acest lucru presupune utilizarea în scop de învățare a sistemului de asistență a aplicațiilor de prelucrare a imaginilor şi a diverselor manuale digitale publicate pe Internet. De asemenea, învățarea prin acțiune presupune elaborarea produselor ce conțin imagini digitale şi publicarea acestora în spațiul virtual.

4.1. Imagini digitale

Cunoaştem deja că în funcție de modul de reprezentare a imaginilor în memoria calculatorului deosebim **grafica orientată pe puncte** şi **grafica orientată pe obiecte**.

Grafica orientată pe puncte

Amintim că în grafica orientată pe puncte imaginile se reprezintă în memoria calculatorului prin împărțirea ei în microzone, denumite *puncte* sau *pixeli*. Descompunerea imaginii în pixeli se realizează cu ajutorul unui *rastru* (de la cuvântul latin *raster*, literalmente "greblă").

În procesul digitalizării imaginii, pixelii sunt parcurşi în ordinea în care se citesc: de la stânga la dreapta, de sus în jos. Fiecărui pixel i se pune în corespondență câteva numere binare, ce reprezintă într-o formă înțeleasă de calculator informațiile despre luminozitatea şi culoarea acestuia. Evident, în grafica orientată pe puncte imaginea este reprezentată printr-o secvență de numere binare, iar toate prelucrările acesteia se realizează prin operații asupra numerelor respective.

Cel mai des, imaginile cu rastru se utilizează pentru prelucrarea informației vizuale obținute din realitatea înconjurătoare. De obicei, majoritatea echipamentelor destinate digitalizării imaginilor, cum ar fi camerele de luat vederi, camerele video, scanerele etc., furnizează la ieşire imagini cu rastru, mai exact fişiere în formate special elaborate pentru reprezentarea, memorarea şi prelucrarea imaginilor cu rastru. De asemenea, imaginile cu rastru sunt utilizate în cazul afişării informațiilor vizuale pe ecranele dispozitivelor digitale şi pentru tipărirea acestora.

Principalul avantaj al imaginilor cu rastru constă în fidelitatea lor, adică în precizia, exactitatea prezentării sau reproducerii realității. În calitate de neajunsuri vom menționa volumul mare de memorie necesar pentru stocarea imaginilor cu rastru şi înrăutățirea calității în cazul măririi acestora.

Grafica orientată pe obiecte

Pentru a asigura o calitate mai bună a imaginilor digitale în cazul redimensionării acestora şi reducerii volumului de memorie necesară pentru stocarea lor, în grafica orientată pe obiecte imaginile sunt formate din obiecte grafice simple: linii, pătrate, dreptunghiuri, circumferinţe, elipse etc.

În calculator, fiecare obiect grafic din componenţa imaginii este codificat printr-un set de numere binare, denumit convenţional **vector**. Un astfel de set de numere binare conţine toate informaţiile necesare pentru desenarea obiectului respectiv: coordonatele centrului şi raza fiecărui cerc, coordonatele vârfului fiecărui dreptunghi etc. Evident, în vectorul ce caracterizează un obiect grafic se includ şi informaţii despre culoarea şi grosimea liniilor ce-l formează, culoarea de umplere, gradientele de culoare etc.

Prelucrarea imaginilor vectoriale se realizează prin recalcularea coordonatelor şi dimensiunilor fiecărui obiect grafic din componenţa acestora cu ajutorul formulelor din geometria analitică. Este important să ştim că în geometria analitică, spre deosebire de geometria tradiţională, figurile sunt definite nu prin imagini, ci cu ajutorul unor formule, iar transformarea acestora se face pur algebric. În acest scop, planul, în cazul imaginilor bidimensionale, şi spaţiul, în cazul imaginilor tridimensionale, sunt dotate cu sisteme de coordonate, de obicei, carteziene.

Imaginile vectoriale pot fi mărite şi micşorate fără a înrăutăţi calitatea acestora, recalculând dimensiunile acestora conform formulelor matematice asociate fiecărui obiect grafic. Mai mult decât atât, cu ajutorul calculelor, obiectele respective pot fi animate, deplasate în spaţiu, recolorate, transfigurate etc., constituind avantaje deosebit de importante în crearea simulatoarelor digitale şi a jocurilor de calculator. Bineînţeles, în cazul aplicaţiilor de grafică digitală, componentele de geometrie analitică pe care se bazează prelucrarea imaginilor vectoriale, adică formulele şi algoritmii de calcul, sunt "invizibile" pentru utilizator, el operând doar cu termeni specifici designului grafic.

Neajunsul imaginilor vectoriale constă în faptul că codificarea informaţiilor vizuale complexe prin obiecte grafice simple, descrise cu ajutorul unor formule matematice, duce la scăderea fidelităţii prezentării sau reproducerii lumii reale. Desigur, scăderea fidelităţii nu este cauzată de formulele matematice propriu-zise, ci de complexitatea şi numărul mare de obiecte ce ar fi necesare pentru a asigura nivelul dorit de fidelitate. Însă, odată cu creşterea performanţelor calculatoarelor moderne, complexitatea şi numărul obiectelor grafice nu mai reprezintă un obstacol de neînvins, fapt ce poate fi uşor observat în cazul jocurilor de calculator, imaginile din componenţa cărora devin tot mai complexe şi tot mai aproape de cele ce le vedem în lumea reală.

Aplicaţiile moderne de prelucrare a imaginilor permit gruparea obiectelor, oferindu-i astfel utilizatorului posibilitatea să creeze din obiecte grafice simple obiecte complexe.

Conversia imaginilor

Imaginile cu rastru pot fi transformate în imagini vectoriale şi invers, cele vectoriale – în imagini cu rastru. În plus, aplicaţiile moderne de procesare a imaginilor permit crearea şi procesarea imaginilor digitale mixte, adică a imaginilor ce conţin atât componente cu rastru, cât şi componente vectoriale.

De obicei, transformarea imaginilor cu rastru în imagini vectoriale se face în scopul micşorării volumului ocupat de ele, al combinării lumii reale cu elemente din lumea

virtuală (realitatea augmentată), al recunoaşterii formelor. Transformarea imaginilor vectoriale în imagini cu rastru se face în vederea afişării acestora pe ecranele echipamentelor digitale şi pentru a fi tipărite. Termenii utilizaţi pentru astfel de transformări sunt **rasterizare** (de la cuvântul *raster*) şi **rendering** (de la cuvântul englez *rendering* "a reda, a exprima în formă vizuală").

4.2. Modele de culori

Pentru captarea imaginilor şi redarea acestora cu ajutorul echipamentelor digitale, se utilizează mai multe modele de culori:

Modelul monocrom, de obicei alb-negru, în care imaginea digitală conţine doar nuanţele de gri ale fiecăreia dintre microzone. În majoritatea cazurilor, aceste nuanţe sunt codificate prin numere binare, formate din opt biţi.

Cantitatea de informaţie într-o astfel de imagine se determină cu ajutorul formulei:

$$I = X\ Y \text{ (octeţi)},$$

unde X şi Y sunt dimensiunile imaginii în puncte.

Modelul *RGB*. Acest model este utilizat pentru a reda imaginile prin emisie de lumină. Ochiul omului, captând lumina emisă de sursa de imagine, percepe culoarea prin "adunarea" celor trei culori componente: roşu (*Red*), verde (*Green*) şi albastru (*Blue*).

Luminozitatea şi culoarea fiecăreia dintre microzone este codificată prin proporţiile celor trei culori de bază. Astfel, în imaginea digitală, fiecărei microzone îi corespund trei octeţi, primul reprezentând nuanţele de roşu, al doilea – nuanţele de verde şi al treilea – nuanţele de albastru.

Cantitatea de informaţie a unei imagini color se determină cu ajutorul formulei:

$$I = 3\ X\ Y \text{ (octeţi)}.$$

Modelul *RGB* se foloseşte în cazul imaginilor ce sunt afişate pe ecranele vizualizatoarelor şi proiectoarelor multimedia. Amintim că în aceste dispozitive fiecare pixel al matricei ce generează imaginea este format din trei celule luminiscente, una dintre ele emiţând culoarea roşie, alta – culoarea verde şi a treia – culoarea galbenă.

De asemenea, modelul *RGB* se foloseşte în camerele de luat vederi, camerele video, în scanere etc. În aceste aparate fiecare pixel al matricei ce captează imaginea este format din trei celule sensibile la lumină, una dintre ele fiind sensibilă doar la lumina roşie, alta – doar la lumina verde şi cea de a treia – doar la lumina galbenă.

Pentru exemplificare, în *figura 4.1* este prezentată fereastra de dialog **Edit color** (Editează culoarea) a aplicaţiei **Paint 3D**, destinată selectării culorii dorite în baza modelului *RGB*.

Modelul *CMY*. Acest model este utilizat pentru redarea imaginilor ce vor fi tipărite. Amintim că imaginile tipărite nu emit, ci doar reflectă lumina care cade pe ele. În consecinţă, în cazul imaginilor tipărite, toată gama de culori este obţinută prin combinaţia vopselelor de trei culori de bază: cyan (*Cyan*), purpuriu (*Magenta*) şi galben (*Yellow*).

Întrucât pentru redarea culorii negre se cere combinaţia tuturor celor trei culori de bază, iar vopselele respective sunt scumpe, în dispozitivele de imprimat se foloseşte

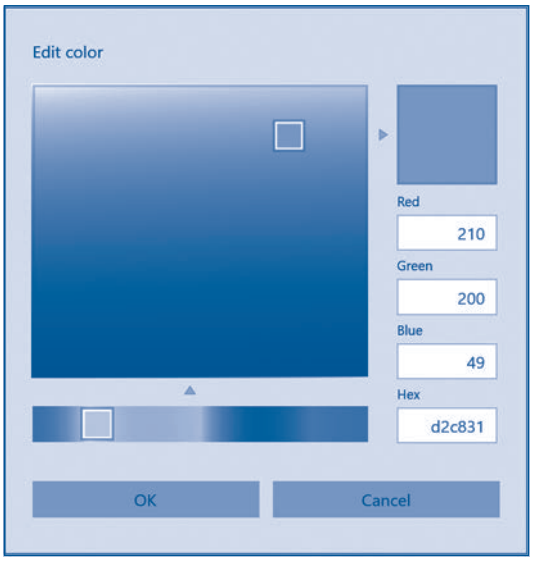


Fig. 4.1. Selectarea culorii dorite în baza modelului *RGB*

şi vopseaua neagră. În consecință, editoarele grafice pun la dispoziția utilizatorului şi modelul de culori *CMYK*, în care, suplimentar la culorile din modelul *CMY*, apare a patra culoare, cea neagră (*black*).

De exemplu, în aplicația **Gimp**, utilizatorul poate alege culoarea dorită cu ajutorul unor cursoare ce setează proporția fiecăreia dintre culorile *CMYK* (*fig. 4.2*). Menționăm faptul că proporțiile respective se indică în procente.

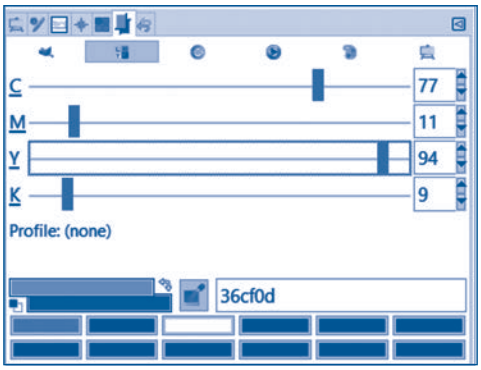


Fig. 4.2. Selectarea culorii dorite în baza modelului *CMYK*

Modele bazate pe palete de culori. Vizual, paleta de culori reprezintă o serie de dreptunghiuri colorate, numerotate, de exemplu, de la 0 la 255. Astfel de palete întâlnim foarte des în industria de înfrumusețare, în care culoarea vopselelor de păr sau a rujurilor pentru buze se indică nu doar prin imagini, dar şi prin numere. Paletele de culori se utilizează pe larg în reparația caroseriilor de automobile, fiecare producător auto oferind paletele respective atelierelor de reparații. De asemenea, paletele de culori se utilizează pe larg în poligrafie, industria textilă, industria vopselelor, designul tehnic ş.a.m.d.

În memoria calculatorului, paletele de culori sunt reprezentate prin codurile culorilor dreptunghiurilor respective conform modelului *RGB* sau *CMY* şi numerele atribuite acestora. Evident, codurile şi numerele respective sunt "invizibile" pentru utilizator, el operând pe ecran doar cu culorile propriu-zise (*fig. 4.3*).

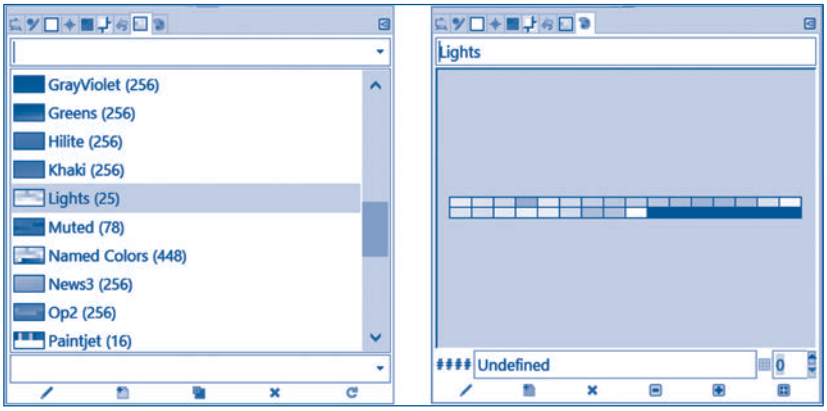


Fig. 4.3. Paleta de culori **Lights** (Lumini) din editorul grafic **Gimp**

În cazurile în care imaginile de prelucrat conţin un număr relativ mic de culori, utilizarea paletelor duce la micşorarea volumului de memorie, necesar pentru stocarea imaginilor respective. Pentru exemplificare, amintim indicatoarele şi marcajele rutiere, panourile publicitare, afişele, posterele, foile volante, schemele diverselor dispozitive tehnice etc.

4.3. Formate de fişiere grafice

Pe parcursul dezvoltării tehnologiilor informaţiei şi comunicaţiilor au fost elaborate mai multe formate de fişiere grafice. Elaborarea de noi şi noi formate grafice a urmărit scopul de a micşora mărimea fişierelor respective, asigurând totodată calitatea imaginilor şi posibilitatea de a le reda pe diverse echipamente.

Comprimarea imaginilor digitale

În general, cantitatea de informaţie din imagini este cu mult mai mare decât cea din texte. De exemplu, imaginile afişate pe ecranele vizualizatoarelor de ultima generaţie au dimensiunile de 8192×4320 de pixeli. Cantitatea de informaţie într-o astfel de imagine este

$$I = 3 \times 8192 \times 4320 = 106168329 \text{ } Octeţi \approx 101 \text{ } Megaocteţi.$$

Matricele fotosensibile ale aparatelor digitale fotografice profesioniste au dimensiuni de 38000×38000 de pixeli. Cantitatea de informaţie într-o imagine captată de o astfel de matrice este

$$I = 3 \times 38000 \times 38000 \approx 4131 \text{ } Megaocteţi \approx 4{,}0 \text{ } Gigaocteţi.$$

Pentru a reduce volumul de memorie necesar în vederea stocării imaginilor digitale, se efectuează comprimarea acestora. În acest scop, au fost elaborați algoritmi speciali, care se studiază în cursurile avansate de informatică. Aceşti algoritmi pot asigura comprimarea imaginilor digitale de zeci şi chiar de sute de ori, cu sau fără pierderea informațiilor grafice.

Algoritmii care comprimă imaginile digitale *fără pierderi de informații* se bazează pe faptul că în multe imagini există microzone adiacente (vecine) identice. Informațiile despre fiecare dintre astfel de microzone pot fi înlocuite cu informația doar despre una dintre ele.

Algoritmii care comprimă imaginile digitale *cu pierderi de informații* reduc numărul de culori din imaginile supuse comprimării şi elimină detaliile foarte mici din ele. În anumite cazuri, aceste transformări nu sunt percepute de ochiul omului.

Formate pentru imaginile de tip rastru

Principalele formate de fişiere grafice pentru imaginile de tip rastru sunt:

BMP – Hartă de biți (*Bitmap*). Acest format este folosit pentru stocarea imaginilor de tip rastru, fiecărei microzone corespunzându-i de la unul (imagini alb-negru) până la 64 de biți (imagini cu o gamă foarte mare de culori). Formatul asigură o calitate foarte bună a imaginilor, însă fişierele respective sunt foarte mari. Nu este recomandat pentru trimiterea prin Internet.

Extensiunile de fişier pentru acest format sunt **.bmp**, **.dib**, **.rle**.

JPEG – acronimul organizației Grupul Comun de Experți în Fotografie (*Joint Photographic Experts Group*). Este un format care permite comprimarea imaginilor de tip rastru cu sau fără pierderi. În cazul comprimării fără pierderi, mărimea fişierelor se micşorează de aproape două ori, iar în cazul comprimării cu pierderi – până la 80 de ori. Este recomandat pentru stocarea fotografiilor şi mai puțin pentru desenele tehnice. Se utilizează pe larg pe Internet.

Extensiile de fişier pentru acest format sunt **.jpg**, **.jfif**, **.jpe** sau **.jpeg**, cea mai populară fiind **.jpg**.

TIFF – acronimul denumirii Format de Fişier de Imagine Etichetată (*Tagged Image File Format*). Este folosit pentru stocarea imaginilor de tip rastru cu o gamă foarte mare de culori, în special a celora din poligrafie. Asigură comprimarea cu sau fără pierderi.

Extensiile de fişier pentru acest format sunt **.tiff** sau **.tif**.

GIF – acronimul denumirii Format pentru Schimb Reciproc de Grafică (*Graphics Interchange Format*). Foloseşte o paletă cu 256 de culori şi asigură comprimarea fără pierderi. Pe o perioadă foarte mare de timp a fost cel mai răspândit format de tip rastru pe Internet. Este foarte bun pentru reprezentarea schemelor, logourilor, siglelor, textelor pe desene şi, mai puțin, pentru fotografii.

Fişierele stocate în acest format au extensiunea **.gif**.

PNG – acronimul denumirii Grafică Portabilă de Rețea (*Portable Network Graphics*). Este destinat stocării imaginilor de tip rastru, comprimate fără pierderi. Formatul PING a fost conceput pentru a înlocui formatul GIF şi, parțial, formatul TIFF. El se foloseşte pe larg pe Internet, fişierele respective având extensia **.png**.

Formate pentru imaginile vectoriale

În cazul imaginilor vectoriale, principalele formate utilizate în prezent sunt:

WMF – acronimul denumirii Metafişier Windows (*Windows Metafile*).

SVG – acronimul denumirii Grafică Vectorială Scalabilă (*Scalable Vector Graphics*).

EPS – acronimul denumirii PostScript Încapsulat (*Encapsulated PostScript*). Limbajul PostScript este pe larg utilizat în edituri.

Uneori, formatele pentru imaginile vectoriale conțin prefixul **2D** sau **3D**, care indică faptul că imaginea este, respectiv, bi- sau tridimensională.

Formate pentru documentele mixte

PDF – acronimul denumirii Format de Document Portabil (*Portable Document Format*). Inițial a fost conceput pentru difuzarea documentelor în formă electronică, în special a celor destinate tipografiilor. Ulterior a căpătat o largă răspândire pe Internet, întrucât nu depinde de specificul sistemelor de operare şi al echipamentelor periferice utilizate pentru tipărirea sau vizualizarea documentelor respective. Permite inserarea într-un singur document a textelor, imaginilor de tip raster şi a imaginilor vectoriale.

CGM – acronimul denumirii Metafişier de Grafică Computațională (*Computer Graphics Metafile*). Se utilizează, în principal, în domeniul proiectării asistate de calculator a celor mai diverse sisteme tehnice şi, mai puțin, în arta digitală sau pe Internet.

De obicei, în aplicațiile destinate prelucrării imaginilor, utilizatorul alege formatul de fişier grafic cu ajutorul unor casete de dialog dedicate (*fig. 4.4*).

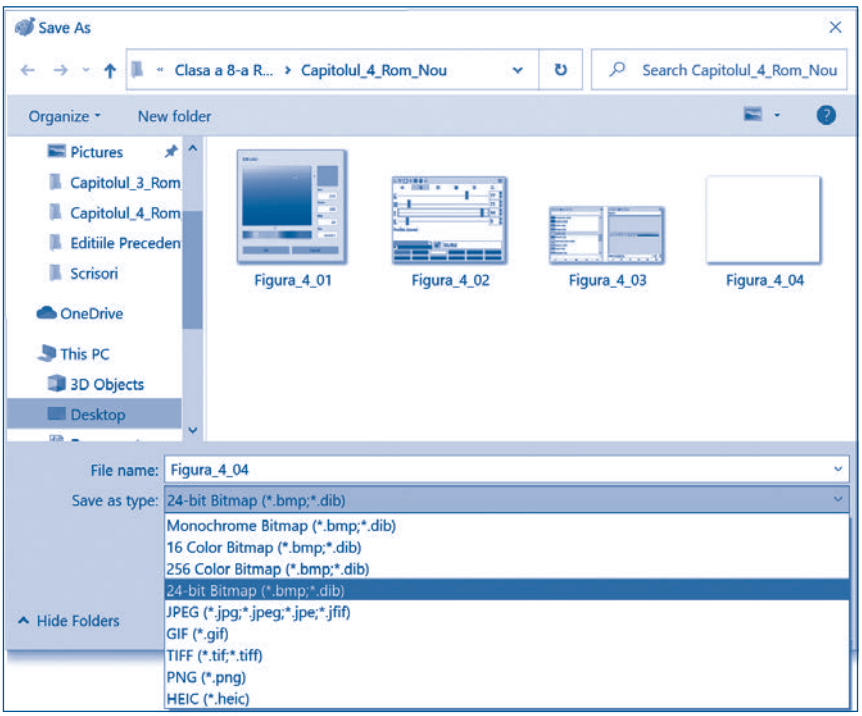


Fig. 4.4. Selectarea formatului de fişier grafic

Accentuăm faptul că multe dintre formatele de fişiere grafice, frecvent utilizate în grafica pe calculator, aparțin unor mari companii, specializate în producerea echipamentelor şi aplicațiilor destinate prelucrării imaginilor digitale.

4.4. Activităţi practice: Echipamente şi formate de fişiere grafice

În scopul consolidării cunoştinţelor teoretice, însuşite în procesul studierii paragrafelor precedente, vă recomandăm următoarele activităţi practice:

1. Recapitulaţi materiile referitoare la cuantizarea imaginilor: microzonă, punct, pixel, rastru, putere de rezoluţie, imagini numerice.

2. **EXPERIMENTEAZĂ!** Aflaţi parametrii tehnici ale camerelor de luat vederi de care dispuneţi: aparatele fotografice digitale, camerele fotografice încorporate în telefoanele mobile şi în tablete. Aflaţi formatele de fişiere grafice utilizate în ele.

3. **CERCETEAZĂ!** Estimaţi dimensiunile şi cantitatea de informaţie în imaginile preluate cu ajutorul camerelor de luat vederi de care dispuneţi.

4. **EXPERIMENTEAZĂ!** Aflaţi parametrii tehnici ale scanerelor, plotterelor şi imprimantelor color de care dispuneţi. Aflaţi formatele de fişiere grafice utilizate în ele.

5. **EXPLOREAZĂ!** Selectaţi din produsele tipografice de care dispuneţi (manuale, dicţionare ilustrate, albume cu fotografii, pliante, foi volante etc.) câteva imagini. Estimaţi cantitatea de informaţie din fiecare dintre aceste imagini.

6. **CREEAZĂ!** Scrieţi un mic studiu despre evoluţia formatelor de fişiere grafice.

7. **STUDIU DE CAZ!** Elaboraţi o analiză comparată a formatelor de fişiere grafice. Folosind un editor grafic, salvaţi una şi aceeaşi imagine în diferite formate. Identificaţi legătura dintre calitatea imaginilor digitale stocate în diferite formate grafice şi volumul de memorie ocupat de acestea.

4.5. Aplicaţia *Paint 3D*

În clasele precedente aţi studiat şi, ulterior, aţi utilizat foarte des aplicaţia **Paint**. Fiind elaborată odată cu apariţia primelor sisteme de operare din familia **Windows**, această aplicaţie simplă s-a dovedit a fi foarte utilă pentru prelucrări elementare ale imaginilor bidimensionale: inserarea şi decuparea fragmentelor, rotirea şi redimensionarea, umplerea cu culoare, inserarea de texte scurte etc.

Odată cu dezvoltarea graficii pe calculator, pentru utilizatorii începători a fost elaborat un nou editor grafic, denumit **Paint 3D**. Sufixul **3D** din denumirea acestei aplicaţii indică faptul că pot fi create şi prelucrate cu ajutorul ei nu doar imaginile bidimensionale, dar şi cele tridimensionale. Ferestra aplicaţiei **Paint 3D** este prezentată în *figura 4.5* (p. 134).

Interacţiunea utilizatorului cu aplicaţia **Paint 3D** se efectuează cu ajutorul elementelor de control caracteristice interfeţelor grafice moderne: pictogramelor, butoanelor, ferestrelor de dialog, panourilor, meniurilor contextuale ş.a.m.d. Astfel, pictogramele din partea de sus a ferestrei au următoarea semnificaţie:

Menu (Meniul) – conţine comenzile destinate deschiderii şi salvării fişierelor grafice. În cazul deschiderii unor imagini bidimensionale (formatele BMP, JPG,

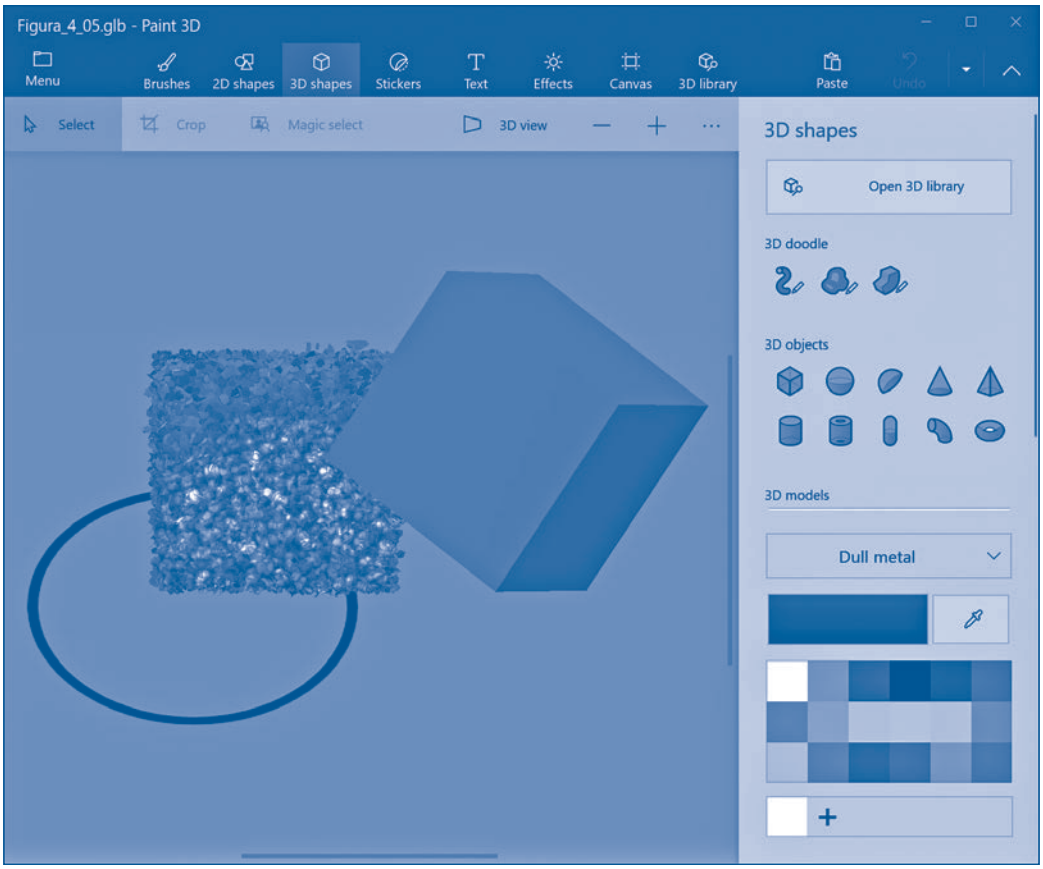


Fig. 4.5. Fereastra aplicației **Paint 3D**

PNG ş.a.m.d.), imaginea de tip raster va servi în calitate de fundal. Orice selecție din această imagine poate fi transformată într-un obiect grafic tridimensional cu ajutorul comenzii **Make 3D** (Transformă într-un obiect 3D) din meniul contextual. Evident, asupra imaginilor de tip raster pot fi efectuate toate operațiile tradiționale: inserarea, decuparea, rotirea, redimensionarea, umplerea cu culoare, inserarea de texte ş.a.m.d.

Imaginile create cu ajutorul aplicației **Paint 3D** pot fi salvate în formate de imagini bidimensionale (opțiunea **Image** din meniul **Save As**), modele 3D (opțiunea **3D** model) sau proiecte (opțiunea **Paint 3D project**). Accentuăm faptul că, pentru stocarea modelelor tridimensionale, au fost elaborate formate grafice speciale, principalele din ele fiind **GLB**, **FBX**, **3MF**).

Pentru a studia în mod individual sau în echipă modul de utilizare a aplicației **Paint 3D**, meniul **Menu** conține o comandă specială **Learn and Feedback** (Află şi retroacționează), care oferă acces la un set de tutoriale online.

Brushes (Pensule) – panoul afişat imediat după lansarea acestei comenzi conține un bogat set de pensule şi o paletă de culori. De asemenea, utilizatorul poate alege aspectul ce va fi atribuit obiectului în curs de desenare, de exemplu mat, luciu, metal nelucios, metal poleit.

2D shapes (Figuri bidimensionale) – panoul respectiv conține figuri predesenate ce pot fi inserate în imagine ca obiecte grafice. Ca oricare alt obiect grafic, utilizatorul are posibilitatea să le seteze proprietățile acestora (grosimea şi culoarea liniilor, culoarea de umplere), să le redimensioneze şi să le rotească.

3D shapes (Figuri tridimensionale) – conține figuri tridimensionale. În afară de figurile predesenate, panoul respectiv conține şi instrumente de desenare, atât a figurilor cu colțuri ascuțite, cât şi a celora cu colțuri rotunjite. Aceste instrumente sunt grupate sub denumirea **Doodle**. Sensul inițial al acestui cuvânt englez este "mâzgălitură", însă în cazul editoarelor grafice, el ar trebui tradus ca "desenarea de mână".

Comanda **Open 3D library** (Deschide librăria 3D) din acest meniu oferă acces la o bogată colecție de modele 3D, stocate pe un server specializat al Corporației Microsoft (*fig. 4.6*).

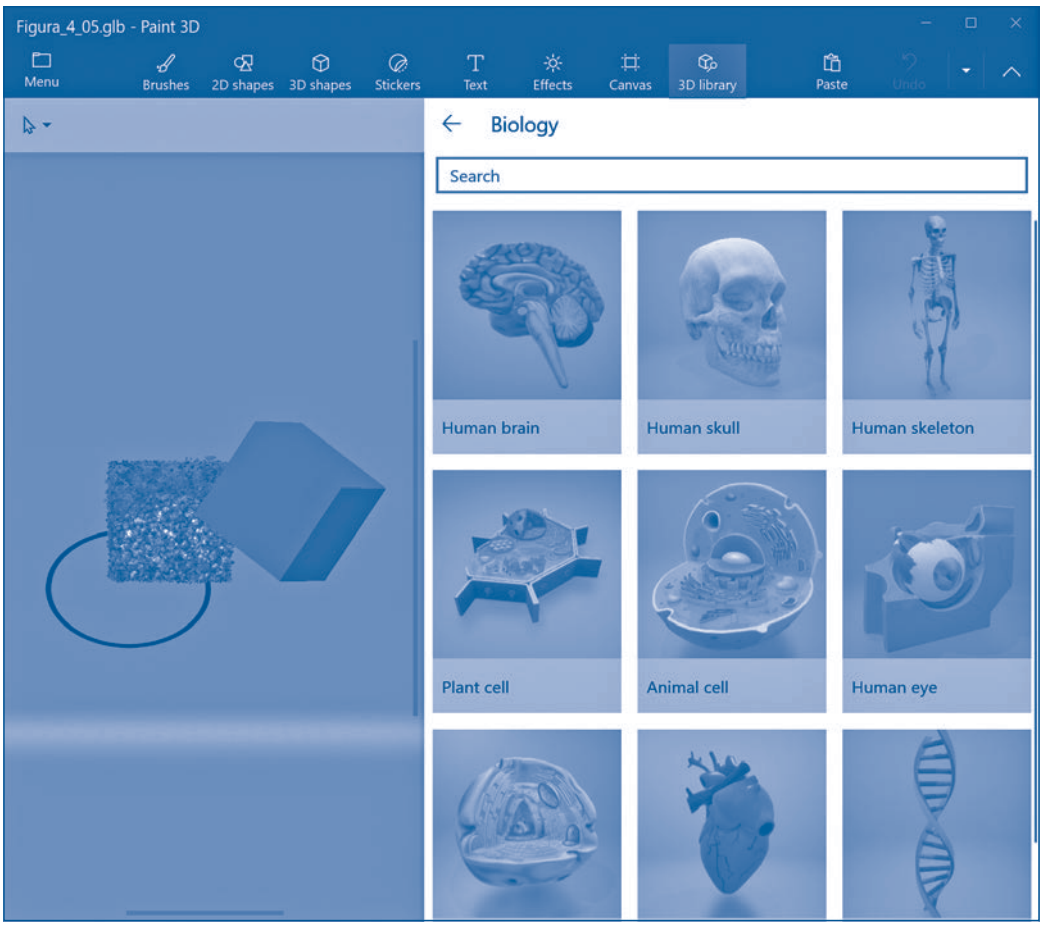


Fig. 4.6. Biblioteca 3D, categoria *Biologie*

Modelele din această bibliotecă sunt grupate pe categorii, principalele dintre ele fiind *Animale*, *Spațiul cosmic*, *Dinozaurii*, *Flori şi plante*, *Biologia* şi multe altele. Evident, utilizatorul poate insera obiectele respective în propriile lucrări grafice,

modificându-le aspectul în funcție de mesajele pe care dorește să le transmită.

Stickers (Autocolante sau Abțibilduri) – oferă accesul la panoul ce conține o colecție bogată de mici etichete, care, fiind atașate unui obiect grafic, devin parte componentă a suprafeței acestuia (*fig. 4.7*).

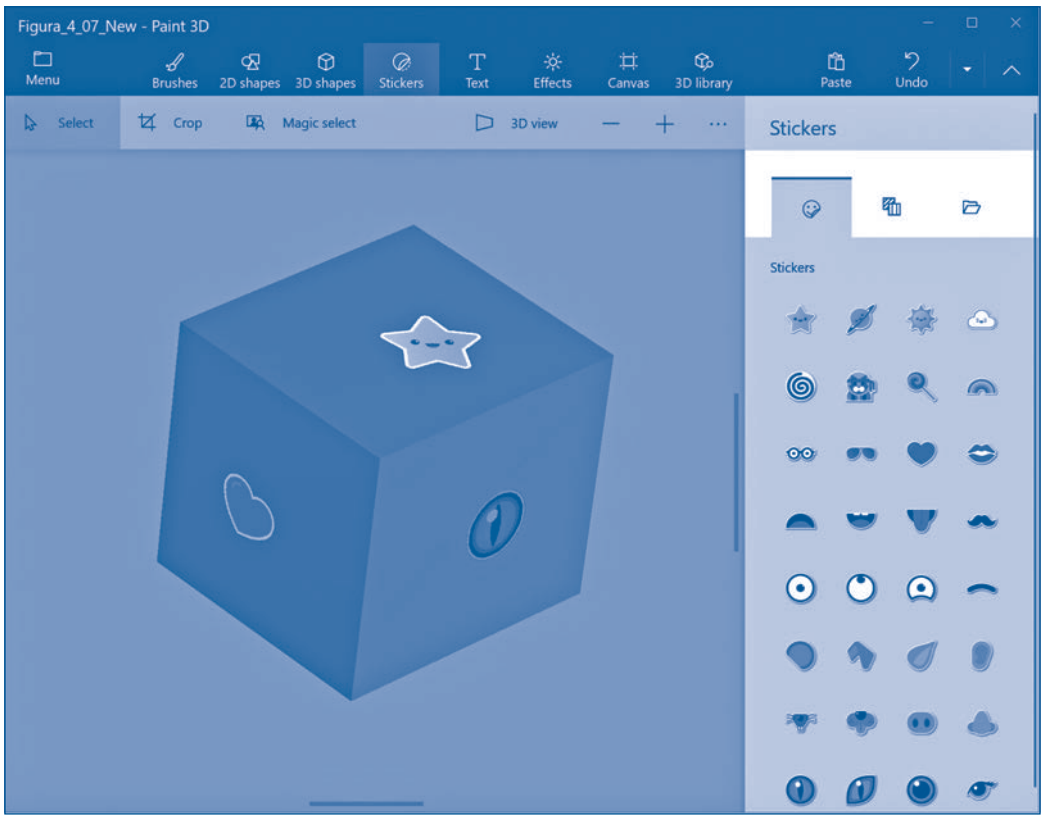


Fig. 4.7. Inserarea autocolantelor

În afară de etichetele predesenate, panoul **Stickers** conține pagina **Textures** (texturi), care oferă suprafeței selectate a obiectului un anumit aspect, de exemplu de lemn, gard viu, blană, pietriș, beton, nisip etc.

Effects (Efecte) – afișează pe ecran un panou din care utilizatorul poate alege filtrele și poziția sursei de lumină.

Din bara de instrumente vom menționa butonul **Magic Select** (Selectarea Magică), care permite selectarea din imaginea de fundal a unei porțiuni ce trebuie eliminată. Fragmentul eliminat este transformat într-un obiect grafic, iar spațiul rămas liber este completat în mod automat cu o imagine generată de inteligența artificială a aplicației **Paint 3D**.

Butonul **3D view** din bara de instrumente oferă utilizatorului posibilitatea să vizualizeze obiectele tridimensionale sub orice unghi, să le rotească, să le schimbe dimensiunile și poziția.

Accentuăm faptul că unele dintre comenzile și meniurile din aplicația **Paint 3D**, de exemplu meniul **Text**, sunt similare celor din aplicația **Paint** sau din grafica orientată pe obiecte ale aplicațiilor de oficiu **Word** și **PowerPoint**.

4.6. Activităţi practice: Prelucrarea imaginilor tridimensionale

1. Creaţi imaginile din zonele de desenare prezentate în *figurile 4.5*, *4.6* şi *4.7*.

2. Încărcaţi în calculator imaginile captate cu ajutorul aparatelor digitale fotografice. Inseraţi în aceste imagini texte scurte ce redau tematica acestora, locul în care au fost preluate, data şi ora. Daţi acestor texte un aspect artistic.

3. Inseraţi în imaginile preluate din aparatele digitale fotografice autocolante sugestive.

4. **ELABOREAZĂ!** Utilizând biblioteca de obiecte tridimensionale, creaţi mici planşe pentru disciplinele reale pe care le studiaţi: matematică, fizică, biologie, chimie, informatică.

5. **CREEAZĂ!** Elaboraţi în baza bibliotecilor de obiecte câteva ilustraţii pentru cele mai captivante opere literare de ficţiune pe care le-aţi citit.

6. **STUDIU INDIVIDUAL SAU ÎN ECHIPĂ!** Parcurgeţi tutorialele video ce pot fi accesate cu ajutorul comenzii **Learn and Feedback** din meniul **Menu** al aplicaţiei **Paint 3D**. Urmăriţi postările *Paint 3D* din *Windows Blog*.

7. **EXERSEAZĂ!** Utilizând ghidurile online de pe Internet, de exemplu cele oferite de serviciul de stocare a materialelor audio şi video **YouTube**, aflaţi destinaţia şi modul de utilizare a fiecăreia dintre comenzile aplicaţiei **Paint 3D**. Creaţi imagini similare celor prezentate în ghidurile respective.

8. **EXPLOREAZĂ!** Găsiţi pe Internet modele de obiecte tridimensionale. Evaluaţi posibilităţile de utilizare a acestora în imaginile pe care intenţionaţi să le elaboraţi. Apreciaţi valoarea artistică a acestora şi relevanţa lor pentru mesajele pe care intenţionaţi să le transmiteţi.

9. **ÎNVAŢĂ SĂ ÎNVEŢI!** Pentru vizualizarea şi modificarea imaginilor tridimensionale, în sistemul de operare **Windows** se utilizează aplicaţia **3D Viewer**. În afară de vizualizarea propriu-zisă, această aplicaţie permite utilizatorului să controleze sursele de iluminare şi să modifice umbrele imaginilor tridimensionale, să aplice diverse texturi. Mai mult decât atât, ea permite crearea imaginilor din categoria realităţii augmentate.

Utilizând sistemul de asistenţă şi ghidurile online, studiaţi în detaliu meniurile şi comenzile acestei aplicaţii. Familiarizaţi-vă cu bogata bibliotecă de modele tridimensionale **3D Library**.

10. **EXPERIMENTEAZĂ!** Obiectele tridimensionale pot fi materializate (tipărite) cu ajutorul imprimantelor 3D. Dacă şcoala dispune de o astfel de imprimantă, tipăriţi câteva dintre obiectele tridimensionale create cu ajutorul aplicaţiei **Paint 3D**.

11. **CERCETEAZĂ!** Elaboraţi un mic studiu al serviciilor de imprimare 3D, disponibile în ţara noastră. Aflaţi cum depinde costul acestui serviciu de tehnologia de imprimare 3D şi complexitatea obiectelor de imprimat.

12. **INTEGREAZĂ!** Aplicaţia **Paint 3D** este foarte utilă pentru elaborarea proiectelor STEAM (*Science*, *Technology*, *Engineering*, *Arts and Mathematics* – Ştiinţă, Tehnologie, Inginerie, Artă şi Matematică). Aceste proiecte se bazează pe integrarea cunoştinţelor din domeniile respective cu ajutorul instrumentelor informatice. Serviciul de stocare a materialelor audio şi video **YouTube** conţine mai multe tutoriale de creare a proiectelor 3D, dedicate integrării cunoştinţelor. Parcurgeţi tutorialele respective şi elaboraţi un proiect STEAM.

4.7. Straturi și canale de culori

Straturi. Prelucrările avansate ale imaginilor digitale presupun utilizarea straturilor. Fiecare strat conține câte o imagine, iar imaginea rezultantă pe care o vede utilizatorul este formată prin suprapunerea acestora (*fig. 4.8*).

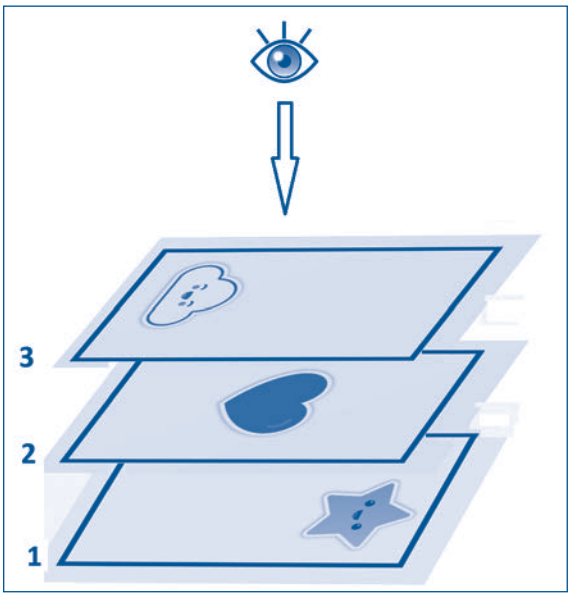


Fig. 4.8. Formarea imaginii din straturi

Evident, imaginea de pe un strat aflat mai jos va fi vizibilă doar atunci când imaginile de pe straturile de mai sus vor fi parțial sau complet transparente. Prin alte cuvinte, un strat poate fi interpretat ca o placă de sticlă cu anumite imagini pe ea.

În cazurile în care unele fragmente ale imaginilor de pe un strat sunt opace (lipsite de transparență), ele vor masca porțiunile respective ale imaginilor ce vin de pe straturile de mai jos. Prin urmare, în cazul utilizării straturilor, fiecărui pixel, pe lângă octeții ce indică proporțiile culorilor de bază, i se mai atașează un octet. Acest octet indică gradul de opacitate, exprimat, de obicei, în procente.

Astfel, gradului de opacitate 0% îi corespunde o transparență completă, imaginea de pe stratul respectiv fiind absolut invizibilă. În acest caz va fi vizibilă în deplină măsură imaginea ce vine de la stratul de mai jos.

Opacității de 100% îi corespunde o netransparență totală, imaginea ce vine de la stratul de mai jos fiind complet eclipsată (acoperită) de imaginea de pe stratul curent.

În cazul gradului de opacitate cu valori intermediare, între 0 şi 100%, imaginea de pe stratul curent se va suprapune pe imaginea ce vine de pe stratul de mai jos, iar odată cu creşterea gradului de opacitate, imaginea de pe stratul curent va atenua tot mai mult imaginea care vine de la stratul de mai jos.

Pentru exemplificare, în *figura 4.9* sunt prezentate imaginile pe care le vede utilizatorul în funcție de gradul de opacitate a straturilor din care ea este formată.

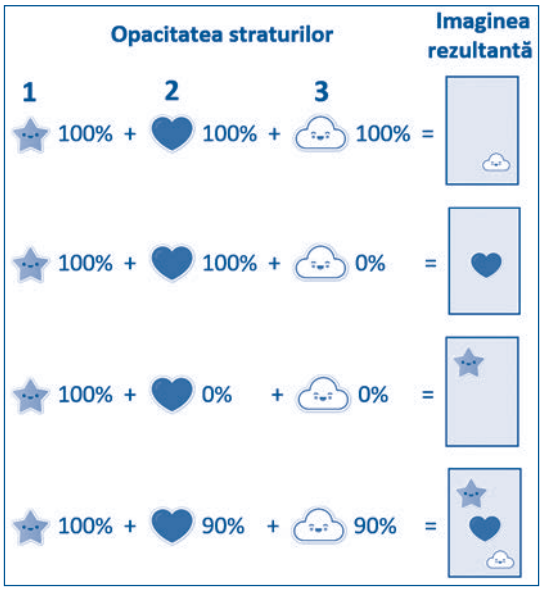


Fig. 4.9. Influența opacității straturilor asupra imaginilor rezultante

În cazul în care Stratul 3, cel de sus, este complet opac, utilizatorul vede doar imaginea de pe el, imaginile de pe celelalte straturi fiind eclipsate. Când Stratul 3 este complet transparent, utilizatorul vede imaginea care vine de la Stratul 2. Întrucât acest strat este complet opac, imaginea ce vine de la Stratul 1 este eclipsată.

În cazul în care Straturile 2 şi 3 sunt complet transparente, utilizatorul vede doar imaginea de pe Stratul 1.

În cazul în care Straturile 2 şi 3 sunt parțial transparente, utilizatorul vede o imagine obținută prin suprapunerea imaginilor de pe toate cele trei straturi. În imaginea rezultantă, cea mai pronunțată este imaginea de pe Stratul 3, care va fi urmată de imaginea de pe Stratul 2, iar cea mai slabă – imaginea de pe Stratul 1.

În ansamblu, utilizarea straturilor permite nu doar simpla suprapunere a mai multor imagini, dar şi aplicarea selectivă a efectelor, mascarea anumitor fragmente de imagini, corectarea eventualelor defecte, de exemplu, cel al "ochilor roşii" de pe fotografiile luate cu bliț.

Canale de culori. Conceptual, un canal de culoare reprezintă un şir de numere, câte unul pentru fiecare dintre pixelii ce formează imaginea. Numărul ataşat fiecăruia dintre pixeli indică proporția culorii asociate canalului în formarea aspectului pixelului.

În momentul încărcării unei imagini, editoarele grafice creează automat trei canale cu denumirile sugestive *Red*, *Green* şi *Blue*, ce corespund culorilor de bază, respectiv, Roşu, Verde şi Albastru. Dacă imaginea este compusă din straturi, editoarele grafice mai creează în mod automat încă un canal, denumit *Alpha*. Acest canal conține informațiile referitoare la transparența/opacitatea fiecăruia dintre pixelii imaginii supuse prelucrării.

În afară de canalele *Red*, *Green*, *Blue* şi *Alpha*, create de editoarele grafice în mod automat, designerul poate crea şi canale proprii, asociindu-le cu culorile dorite.

Pentru exemplificare, în *figura 4.10* (p. 140) sunt prezentate ferestrele de dialog ale editorului grafic **GIMP**, destinate lucrului cu canalele de culoare.

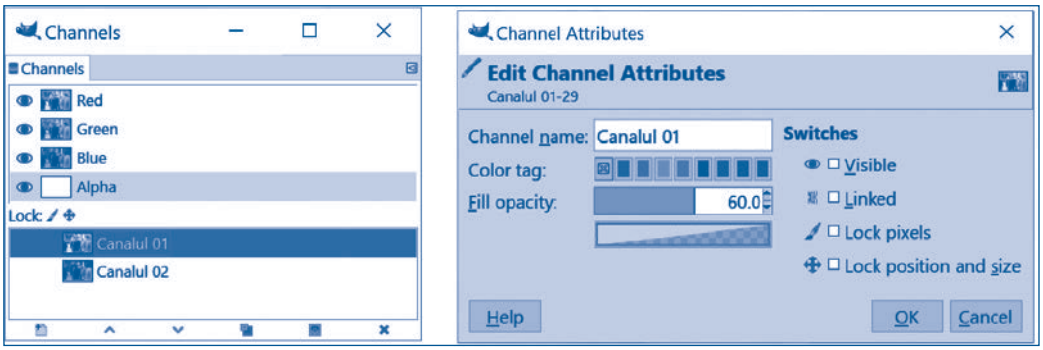


Fig. 4.10. Ferestre de dialog destinate lucrului cu canalele de culoare

Fereastra **Channels** (Canale) conține canalele create în mod automat *Red*, *Green*, *Blue* şi *Alpha* şi canalele create de utilizator *Canalul 01* şi *Canalul 02*. Fereastra **Channel Attributes** (Atributele Canalului) conține elementele de control ce permit setarea şi modificarea proprietăților primului dintre canalele create de utilizator.

Activarea sau dezactivarea unui canal duce la includerea sau, respectiv, excluderea culorii respective din aspectul pixelilor imaginilor rezultante. Activând sau dezactivând anumite canale şi modificând proprietățile acestora, designerul poate corecta culorile imaginilor supuse prelucrării, poate crea măşti pentru includerea, excluderea, accentuarea sau estomparea anumitor fragmente din imaginile respective.

4.8. Aplicația *GIMP*

Editorul grafic **GIMP** (*General Image Manipulation Program* – Program General de Manipulare a Imaginilor) este o aplicație dezvoltată de un grup de voluntari şi este gratuită.

Aplicația **GIMP** poate rula în una sau în mai multe ferestre. De obicei, în cazul rulării în mai multe ferestre, imaginea de prelucrat este afişată în fereastra principală, cea mai mare, pe când celelalte ferestre, numite **dialoguri**, conțin elementele de control necesare pentru a executa cele mai diverse sarcini. Ferestrele de dialog pot fi glisate pe ecranul vizualizatorului, permițând astfel designerului să utilizeze cât mai eficient masa de lucru a sistemului de operare. În cazul graficii pe calculator acest lucru este foarte important, întrucât editoarele respective conțin sute de astfel de ferestre, iar executarea chiar şi a unei sarcini necesită afişarea concomitentă pe ecran a 3-4 şi chiar a mai multor ferestre de dialog.

Fereastra principală a aplicației **GIMP** (*fig. 4.11*) conține, de sus în jos, bara cu denumirea fişierului supus prelucrării, bara de meniuri, zona în care sunt prelucrate imaginile şi bara de stare. Utilizatorul îşi personalizează spațiul de lucru şi prelucrează imaginile cu ajutorul comenzilor din meniurile din bara de meniuri şi al comenzilor din meniurile contextuale.

Meniurile din partea de sus a ferestrei aplicației **GIMP** au următoarea semnificație:

Fig. 4.11. Fereastra aplicației **GIMP**

File – conține comenzile destinate deschiderii, salvării şi închiderii fişierelor grafice. Un fişier grafic poate fi deschis ca o imagine sau ca un strat. În ultimul caz, imaginea din fişierul respectiv va fi inclusă în calitate de un strat distinct al imaginii în curs de prelucrare.

Edit – conține mai multe comenzi, destinate editării imaginilor, cum ar fi tăierea, copierea, lipirea etc. Tot în acest meniu se regăseşte comanda **Preferences** (Preferințe), care permite personalizarea spațiului de lucru.

Select – conține diverse instrumente de selectare, dar şi de salvare a conținuturilor selectate.

View – comenzile din acest meniu permit alegerea atât a modurilor de afişare a imaginilor de prelucrat, cât şi afişarea/ascunderea unor ferestre de dialog.

Image – include o listă de comenzi destinate stabilirii proprietăților generale ale imaginilor şi realizării operațiilor frecvent utilizate în procesul de prelucrare a imaginilor: canavaua, alinierea la grilă, rotirea, oglindirea etc.

Layer – în acest meniu sunt grupate principalele comenzi de lucru cu straturile din care este formată imaginea supusă prelucrării: crearea, eliminarea, duplicarea straturilor, schimbarea ordinii straturilor, setarea opacității/transparenței, crearea măştilor, combinarea straturilor etc.

Colors – include comenzile destinate prelucrării culorilor. Vom menționa din ele comenzile legate de ajustarea luminozității şi a contrastului, de reducere sau de extindere a numărului de culori, de aproximare a culorilor din imagine cu cele mai apropiate dintr-o anumită paletă de culori.

Tools – conține o listă a comenzilor destinate prelucrării imaginilor, grupate pe categorii: de selecție, de desenare, de transformare a imaginilor, de inserare de texte etc. Utilizatorul poate personaliza mediul de lucru, creând aşa-numitele

Toolboxes (Casete de instrumente), în care va grupa pictogramele instrumentelor frecvent utilizate. Aceste casete pot fi afişate permanent pe ecran, fie în cadrul unei singure ferestre, împreună cu imaginea de prelucrat, fie în cadrul unor ferestre separate.

Filters – conține mai multe filtre, care, fiind aplicate asupra imaginii în curs de prelucrare, îi schimbă aspectul acesteia: corectarea "ochilor roşii", estomparea, distorsionarea, îmbunătățirea clarității, aplicarea de lumini şi umbre, includerea perspectivei şi multe altele.

Windows – include comenzile destinate gestiunii ferestrelor. Anume în acest meniu se conține şi comanda care setează modul de rulare a aplicației **GIMP**: în una sau mai multe ferestre. Tot în acest meniu se regăseşte o listă completă a ferestrelor de dialog ce pot fi afişate permanent pe ecran: setul de pensule, paletele de culori, straturile ş.a.m.d.

Help – sistemul de asistență. În funcție de modul în care a fost personalizată aplicația **GIMP**, acest sistem poate fi instalat pe calculatorul local sau accesat la distanță. Menționăm faptul că sistemul de asistență este disponibil nu doar în limba engleză, dar şi în limbile română şi rusă.

> **Aplicația *GIMP* conține sute de comenzi, butoane, cursoare, casete şi alte elemente de control. Niciun manual nu ar putea cuprinde o descriere completă a acestora. Prin urmare, aflați destinația comenzilor şi elementelor de control poziționând cursorul pe fiecare dintre ele sau apelând la sistemul de asistență.**

De obicei, la activarea unei comenzi, pe ecran se afişează o fereastră de dialog, din care utilizatorul poate alege opțiunile dorite. Pentru exemplificare, în *figura 4.12* este prezentată fereastra de dialog **Layers** (Straturi), care este afişată pe ecran imediat după lansarea comenzii **Windows**, **Dockable Dialogs**, **Layers**. Această fereastră conține stiva de straturi şi elementele de control ce permit utilizatorului să seteze opacitatea acestora, să schimbe ordinea în care ele apar în stivă, să le excludă din stiva respectivă, să includă straturi noi.

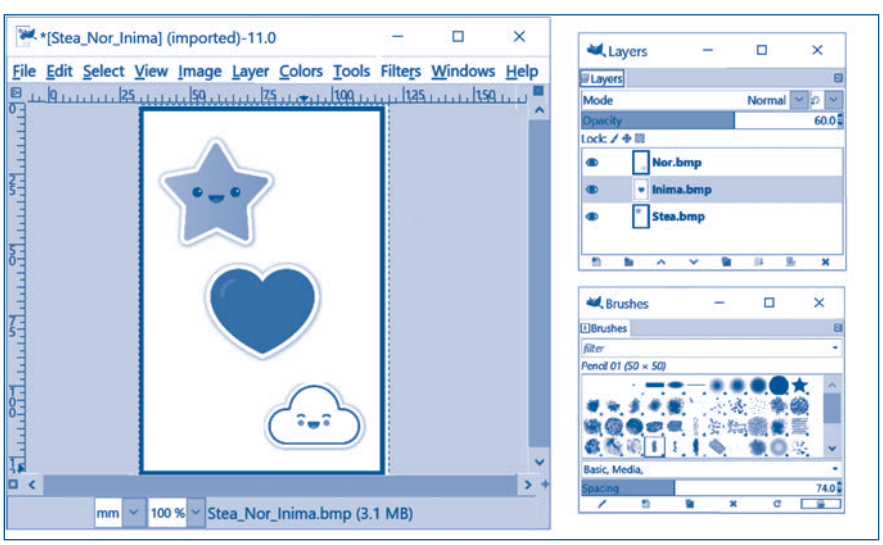


Fig. 4.12. Fereastra principală şi ferestrele de dialog **Layers** şi **Brushes**

Tot pe *figura 4.12* este afişată şi fereastra de dialog **Brushes**, care permite selectarea pensulelor şi setarea parametrilor acestora.

4.9. Activităţi practice: Prelucrarea imaginilor de tip rastru

1. Utilizând comenzile din meniul aplicaţiei **GIMP**, personalizaţi-vă mediul de lucru:
– setaţi modul de lucru în una sau în mai multe ferestre;
– decideţi asupra comenzilor şi instrumentelor pe care intenţionaţi să le utilizaţi mai des şi setaţi modul de afişare permanentă pe ecran a casetelor şi dialogurilor respective;
– racordaţi setările editorului grafic la parametrii tehnici ai calculatorului cu care lucraţi şi ai echipamentelor digitale de care dispuneţi: vizualizatoare, camere fotografice, scanere, imprimante color, proiectoare multimedia.

2. Localizaţi în meniurile aplicaţiei **GIMP** instrumentele necesare pentru realizarea următoarelor operaţii grafice:
– selectarea anumitor fragmente din imaginea supusă prelucrării;
– schimbarea dimensiunilor imaginii;
– tăierea marginilor;
– rotirea şi oglindirea fragmentelor selectate şi a imaginilor în ansamblu;
– inscripţionarea imaginilor;
– compunerea imaginilor din mai multe straturi;
– modificarea culorilor;
– modificarea luminozităţii şi a contrastului;
– aplicarea efectelor;
– comprimarea imaginilor.

3. **EXPERIMENTEAZĂ!** Încărcaţi în calitate de straturi 3-4 imagini. Schimbând opacitatea straturilor şi ordinea acestora în stivă, observaţi cum se modifică imaginea rezultantă.

4. **CREEAZĂ!** Utilizând straturile şi instrumentele de operare cu acestea, creaţi imagini rezultante ce reprezintă animalele exotice pe fundalul mediului lor de viaţă.

5. **EXPERIMENTEAZĂ!** Preluaţi cu ajutorul camerei de luat vederi câteva poze pe cele mai diverse subiecte. Încărcaţi pozele respective în calculator. Utilizând canalele standard de culori, determinaţi cum se modifică aspectul pozelor în cazul activării sau dezactivării acestora.

6. **CERCETEAZĂ!** Utilizând canalele de culori create din proprie iniţiativă, determinaţi cum modificarea atributelor acestora influenţează aspectul imaginilor rezultante.

7. **ÎNVAŢĂ SĂ ÎNVEŢI!** Cea mai eficientă metodă de a studia numeroasele efecte din meniurile aplicaţiei **GIMP** constă în aplicarea lor separată pe diverse imagini. Creaţi-vă o bibliotecă de imagini ce conţine portrete, peisaje, scene sportive, opere de artă plastică, obiecte arhitecturale ş.a.m.d. şi aplicaţi fiecare dintre efectele grafice ale aplicaţiei **GIMP**. Observaţi cum se modifică imaginile supuse prelucrării în funcţie de specificul efectelor grafice aplicate.

4.10. Proiecte recomandate

Lucrând în echipe, elaborați unul sau două dintre proiectele ce urmează:

1. Albume digitale: viața școlii, viața clasei, de vacanță, satul/orașul meu, portrete, natură moartă, reportaje, călătorii, obiecte arhitecturale, peisaje, competiții sportive, animale, artă fotografică abstractă.

2. Arborele genealogic al uneia dintre personalitățile celebre, originare din localitatea natală.

3. Colecții de fotografii digitale didactice pentru:
– diverse discipline școlare;
– muzeul școlii, muzeul satului/orașului natal;
– profilurile personale de pe Internet.

4. Colecții digitale de indicatoare, semne (rutiere, de securitate a muncii, de avertizare, de informare).

5. Expoziții tematice de fotografii digitale.

6. Planșe pentru diverse discipline școlare.

7. Expoziții tematice de grafică digitală.

8. Pliante, postere tematice, placate:
– drepturile copilului;
– activismul civic;
– voluntariatul;
– modul sănătos de viață;
– ecologia și protecția mediului;
– securitatea și siguranța pe Internet;
– temă liberă.

9. Profiluri de personalități ilustre.

10. Reportaje fotografice de la:
– evenimentele festive;
– concursurile școlare;
– evenimentele artistice;
– evenimentele de divertisment;
– activitățile civice;
– activitățile de voluntariat.

11. Texte artistice:
– citate remarcabile;
– citate cu evidențierea mesajului principal;
– mottouri.

În procesul elaborării proiectelor puteți utiliza atât propriile imagini, preluate cu ajutorul camerelor fotografice digitale, cât și imaginile relevante, descărcate de pe Internet. Vă rugăm să respectați cu strictețe regulile de securitate și de siguranță pe Internet, etica digitală, dreptul de autor.