CIOARĂ PAULA-CONSTANTINA

DEZVOLTAREA GÂNDIRII CRITICE, CREATIVE, COMPARATIVE, ANALOGICE A ELEVILOR PRIN DISCIPLINELE INFORMATICĂ ŞI TIC ÎN ÎNVĂŢĂMÂNTUL PRIMAR ŞI GIMNAZIAL

EDITURA EVOMIND

DEZVOLTAREA GÂNDIRII CRITICE, CREATIVE, COMPARATIVE, ANALOGICE A ELEVILOR PRIN DISCIPLINELE INFORMATICĂ ŞI TIC ÎN ÎNVĂŢĂMÂNTUL PRIMAR ŞI GIMNAZIAL

ISBN 978-606-94762-9-1

Editura Evomind, 2019

https://evomind.org/

CUPRINS

INTRODUCERE

„Capul copilului nu este un vas pe care să-l umpli, ci o făclie pe care s-o aprinzi astfel încât, mai târziu, să lumineze cu lumină proprie.” (Plutarh)

Istoria marilor invenţii şi descoperiri, a operelor de artă şi a revoluţiei tehnico-ştiinţifice este istoria inteligenţei şi a creativităţii, darul cel mai de preţ al omului, care i-a permis să făurească primele unelte, să stăpânească natura prin ştiinţă şi tehnică, să creeze un peisaj nou pe planeta noastră şi să pătrundă în spaţiul cosmic. În cuceririle ştiinţei, ale tehnicii şi culturii artistice sunt materializate capacităţile creatoare ale omului, inteligenţa şi sensibilitatea lui faţă de frumos. Bogăţia unui popor nu stă în bani şi nici în milioanele de tone de petrol, ci în muncă şi inteligenţă creatoare, acest aur cenuşiu de nepreţuit al omului.

Progresul omenirii nu este posibil fără activitatea creatoare, teoretică sau practică, a omului. Pentru ridicarea omului pe noi trepte de bunăstare, de cultură şi de civilizaţie, şcoala, ca factor principal în formarea cadrelor necesare în toate sectoarele vieţii, are un rol bine definit privind dezvoltarea la elevi a inteligenţei şi creativităţii.

După unele statistici, nu s-a utilizat până acum decât 3% din inteligenţa omenirii, cauzele mai importante fiind analfabetismul şi concepţia şcolii tradiţionale, bazată îndeosebi pe predarea, memorarea şi reproducerea cunoştinţelor, neglijând dezvoltarea potenţialului creator al elevilor şi însuşirea de către aceştia a unor tehnici de învăţare productivă.

Dacă la începutul secolului nostru accentul era pus, în general, pe identificarea creatorilor potenţiali, în ultimul timp accentul s-a pus pe studiul căilor de formare şi educare a capacităţilor creatoare. Opinii similare exprimă A.F. Osborn, citat de Al. Roşca: „Până nu de mult se credea că o persoană este fie creatoare, fie necreatoare, şi că în această privinţă nu este nimic de făcut. Până acum, cercetarea ştiinţifică a stabilit că aptitudinile creatoare pot fi deliberat şi măsurabil dezvoltate”[1].

Şcoala contemporană, centrată pe elev, are un rol bine precizat în dezvoltarea uriaşului potenţial intelectual, reprezentat de inteligenţă şi creativitate, care, pus în valoare, va asigura neîntrerupt progresul social şi uman.

Caracterizată printr-un înalt grad de complexitate şi printr-un ritm accelerat de dezvoltare, creativitatea a fost apreciată drept o însuşire generatoare a progresului. Astăzi, toate domeniile de

[1] Al., Roşca, *Creativitatea generală şi specifică*, Bucureşti, Editura Academiei, 1981, p. 21.

activitate din societatea noastră cer de la om capacitatea de a gândi şi a acţiona creator, prin proprie iniţiativă şi în mod independent.

Deşi dezvoltarea însuşirilor creative constituie o sarcină de prim ordin a învăţământului de toate gradele, nu există un obiect, o activitate specifică care să urmărească cu predilecţie acest scop.

Şcoala trebuie să fie locul unde talentele sunt recunoscute, stimulate şi dezvoltate. Creativitatea copilului este diferită de creativitatea autentică pe care o întâlnim la un adult, în sensul că produsul activităţii sale creatoare nu este nou. Este însă nou pentru el şi este realizat în mod independent. Activitatea de creaţie autentică a omului matur este momentul final al unui lung proces, în care creaţia copilului se înscrie ca o premisă indispensabilă. Şcoala are posibilitatea de a capta, canaliza şi dezvolta creativitatea elevilor chiar şi fără a interveni modificări radicale în structura şcolii sau în programa de învăţământ.

Învăţarea de tip creativ, cu metodele şi mijloacele ei specifice, poate influenţa climatul creativ al copilului prin ponderea activităţilor instructiv-educative, prin stimularea independenţei şi spontaneităţii creatoare, prin remarcarea unor subiecţi cu o tipicitate creativă, apreciind întotdeauna, promovând şi încurajând efortul creator.

Orice act care reclamă din partea elevilor procedee

euristice şi restructurare, care duce la concluzii individuale inedite, descoperite prin eforturi personale, este un act creator.

La copil se manifestă în mod spontan o curiozitate şi receptivitate vie, imaginaţie bogată, nevoia de succes şi apreciere, tendinţă spre activitate şi investigaţie, particularităţi strâns legate de mobiluri intrinseci ale oricărui act de producţie originală.

Învăţământul tradiţional a fost, în mare măsură, pasiv şi reproductiv. Este foarte important să căutăm metode prin care să dezvoltăm la elevi capacităţi creatoare şi spirit de investigaţie.

În procesul de învăţământ nu interesează produsul elevilor ca valoare socială, ci, în plan psihologic, interesează supleţea soluţiei găsite pentru realizarea problemelor şcolare solicitate de către profesor sau profesor.

Elevul este pus să privească o problemă din unghiuri diferite, s-o interpreteze, să caute independent o soluţie, să acţioneze ca şi când ar descoperi pentru sine acele cunoştinţe care au fost descoperite în procesul dezvoltării istorice a omenirii şi pe care el trebuie să şi le însuşească.

A pune elevul în situaţia de a dobândi cunoştinţe în mod independent, sub conducerea profesorului, înseamnă a organiza în aşa fel procesul de învăţământ încât să constituie un neîntrerupt proces de punere în faţa elevilor de noi şi noi probleme, într-un grad de complexitate crescut. Acest nou tip de învăţământ nu duce

numai la formarea unui stil de abordare creatoare a problemelor, ci şi la educarea unor trăsături de personalitate, îi măreşte încrederea în sine, îi stimulează atenţia şi interesul. Elevul este transformat într-un participant activ la procesul pedagogic, stimulându-i-se motivaţia intrinsecă.

Educaţia menită să ducă la dezvoltarea creativităţii, a gândirii critice este în esenţă o bună educaţie generală, care trebuie avută în vedere pentru toţi elevii.

Pedagogii accentuează tot mai mult că formarea omului priveşte nu numai aspectul intelectual, ci întreaga personalitate. Spiritul creativ, deşi întemeiat pe formarea şi dezvoltarea unor calităţi ale gândirii, nu poate fi numai rezultatul educaţiei intelectuale, ci şi al dezvoltării personalităţii, ceea ce presupune o punere în valoare mai eficientă a posibilităţilor educative şi formative ale procesului de învăţământ.

Se pledează pentru o educaţie în direcţia formării gândirii independente, a punerii juste a problemelor, o educaţie în care îndrăzneala de a gândi nonconformist, curiozitatea şi creativitatea trebuie să fie stimulate.

Am ales această temă pentru lucrarea de gradul I, mai întâi pentru a completa informarea şi documentarea în domeniul informaticii la nivelul claselor mici, apoi pentru a veni sprijinul colegilor mai tineri care predau la aceste clase.

A fost de asemenea util şi interesant să confrunt ideile şi concluziile rezultate din studiul diverselor materiale pe această temă cu propria experienţă la clasă, în scopul creşterii competenţei şi performanţei elevilor în domeniul informaticii.

Am socotit că este necesar, „să adun” în această lucrare un set de metode şi procedee folosite şi validate în activitatea personală pentru creşterea eficienţei lecţiilor de informatică la nivelul primar şi gimnazial.

CAPITOLUL 1
ALGORITMI PENTRU DEZVOLTAREA GÂNDIRII CRITICE ŞI CREATIVE

1.1. ASPECTE GENERALE

Gândirea algoritmică s-a transformat azi dintr-un instrument matematic particular într-o modalitate fundamentală de abordare a problemelor în domenii care nu au nimic comun cu matematica. Acest lucru s-a realizat prin interacţiunea algoritm-calculator. Azi, prin algoritm înţelegem o metodă de rezolvare a unui tip de probleme, metodă ce se poate implementa pe calculator. O definiţie acceptată pentru a algoritm este: o mulţime finită, bine determinată de acţiuni (paşi) care aplicată asupra unor date de intrare furnizează un rezultat. Algoritmul trebuie să îndeplinească anumite caracteristici şi principalele caracteristici ale unui algoritm trebuie să fie (potrivit www.scism.sbk.ac.uk/law/Section5/chap1/s5c1p2.html):

- generalitate – algoritmul trebuie să fie cât mai general astfel ca să rezolve o clasă cât mai largă de probleme;
- claritate/determinare – acţiunile algoritmului trebuie să fie clare, simple şi riguros specificate;

- finitudine – acţiunile algoritmului trebuie să se termine după un număr finit de operaţii, aceasta pentru orice set de date valide;

- corectitudine – algoritmul trebuie să producă un rezultat corect (date de ieşire) pentru orice set de date de intrare valide;

- performanţă – algoritmul trebuie să fie eficient privind resursele utilizate, şi anume să utilizeze memorie minimă şi sa se termine într-un timp minim.

Exemple.

1. *Nu orice problemă poate fi rezolvata algoritmic.*

Considerăm un număr natural *n* şi următoarele două probleme:

-să se construiască mulţimea divizorilor lui *n*;

-să se construiască mulţimea multiplilor lui *n*.

Pentru rezolvarea primei probleme se poate elabora uşor un algoritm, în schimb pentru a doua problemă nu se poate scrie un algoritm atâta timp cât nu se cunoaşte un criteriu de oprire a prelucrărilor.

2. *Un algoritm trebuie să funcţioneze pentru orice date de intrare.*

Să considerăm problema ordonării crescătoare a şirului de valori: 2*;* 1*;* 4*;* 3*;* 5. O modalitate de ordonare ar fi următoarea: se compară primul element cu al doilea iar daca nu se află în ordinea

bună se interschimbă în felul acesta se obţine (1*;* 2*;* 4*;* 3*;* 5)); pentru şirul astfel transformat se compară al doilea element cu al treilea şi dacă nu se află în ordinea dorită se interschimbă (la aceasta etapă şirul rămâne neschimbat); se continuă procedeul până penultimul element se compară cu ultimul. În felul acesta se obţine (1*;* 2*;* 3*;* 4*;* 5).

Deşi metoda descrisă mai sus a permis ordonarea crescătoare a şirului (2*;* 1*;* 4*;* 3*;* 5) ea nu poate fi considerata un algoritm de sortare întrucât dacă este aplicată şirului (3*;* 2*;* 1*;* 4*;* 5) conduce la (2*;* 1*;* 3*;* 4*;* 5).

2. *Un algoritm trebuie să se oprească.*

Se consideră următoarea secvenţă de prelucrări:

Pas 1. Atribuie variabilei *x* valoarea 1;

Pas 2. Măreşte valoarea lui *x* cu 2;

Pas 3. Dacă *x* este egal cu 100 atunci se opreşte prelucrarea altfel se reia de la Pas 2.

Este uşor de observat că *x* nu va lua niciodată valoarea 100, deci succesiunea de prelucrări nu se termină niciodată. Din acest motiv nu poate fi considerata un algoritm corect.

3. *Prelucrările dintr-un algoritm trebuie să fie neambigue.*

Considerăm următoarea secvenţă de prelucrări:

Pas 1. Atribuie variabilei *x* valoarea 0;

Pas 2. *Fie* se măreşte *x* cu 1 *fie* se micşorează *x* cu 1;

Pas 3. Dacă $x \in [-10; 10]$ se reia de la Pasul 2, altfel se opreşte algoritmul.

Atât timp cât nu se stabileşte un criteriu după care se decide dacă *x* se măreşte sau se micşorează secvenţa de mai sus nu poate fi considerata un algoritm. Ambiguitatea poate fi evitata prin utilizarea unui limbaj riguros de descriere a algoritmilor. Să considerăm că Pas 2 se înlocuieşte cu:

Pas 2. Se aruncă o monedă. Dacă se obţine cap se măreşte *x* cu 1 iar dacă se obţine pajură se micşorează *x* cu 1;

În acest caz specificarea prelucrărilor nu mai este ambiguă chiar dacă la execuţii diferite se obţin rezultate diferite. Ce se poate spune despre finitudinea acestui algoritm ? Dacă s-ar obţine alternativ cap respectiv pajură, prelucrarea ar putea fi infinita. Exista însă şi posibilitatea să se obţină de 11 ori la rând cap (sau pajură), caz în care procesul s-ar opri după 11 repetări ale pasului 2.

Atât timp cât şansa ca algoritmul să se termine este nenulă algoritmul poate fi considerat corect (în cazul prezentat mai sus este vorba despre un algoritm aleator).

4. *Un algoritm trebuie să se oprească după un interval rezonabil de timp.*

Să considerăm că rezolvarea unei probleme implică prelucrarea a *n* date şi că numărul de prelucrări $T(n)$ depinde de *n*.

Presupunem că timpul de execuţie a unei prelucrări este 10^{-3}s şi că problema are dimensiunea $n = 100$. Dacă se foloseşte un algoritm caracterizat prin $T(n) = n$ atunci timpul de execuţie va fi $100x10^{-3} = 10^{-1}$ secunde.

Dacă, însă se foloseşte un algoritm caracterizat prin $T(n) = 2n$ atunci timpul de execuţie va fi de circa 1027 secunde adică aproximativ 1019 ani.

În afara acestor proprietăţi la scrierea unui algoritm trebuie bine stabilite de la început **datele de intrare** (intrarea algoritmului – datele pe care le avem la dispoziţie prin tema de proiectare – enunţul problemei) şi **datele de ieşire** (ieşirea algoritmului rezultatele pe care vrem să le obţinem). Atât pentru datele de intrare, cât şi pentru datele de ieşire trebuie să stabilim:

- tipul lor (date de tip întreg, real, şiruri de caractere)
- domeniul în care aceste date pot lua valori. (de exemplu dacă se specifică în problemă: se citeşte un număr natural mai mic decât 20, trebuie să verificăm dacă valoarea citită este într-adevăr mai mică decât 20 şi mai mare ca zero – validarea datelor).

Execuţia unui algoritm înseamnă execuţia pas cu pas a operaţiilor (instrucţiunilor) descrise

Întreaga activitate de cercetare şi elaborare de software din domeniul Tehnologiei Informaţiei este determinată de inventarea, conceperea, elaborarea, testarea, şi implementarea de algoritmi

performanţi şi utili. Marea diversitate a algoritmilor şi marea aplicabilitate a acestora în toate domeniile, face ca această temă sa fie mereu actuală şi într-o continuă schimbare şi perfecţionare vezi link-ul (www.cs.pitt.edu/~kirk/algorithmcourses/)

Folosind instrumente ce căutare pe Internet găsim şi alte definiţii:

Potrivit http://mathworld.wolfram.com/Algorithm.html avem următoarea definiţie:

Algorithm = *A specific set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the procedure terminate at some point. Specific algorithms sometimes also go by the name method, procedure, or technique. The word "algorithm" is a distortion of al-Khwarizmi, a Persian mathematician who wrote an influential treatise about algebraic methods.* Exemplele pot continua.

În procesul de rezolvare a anumitor probleme folosind un program scris pe un calculator, într-un mediu de programare, se disting următoarele patru etape:

- descrierea algoritmului
- demonstrarea corectitudinii sale
- analiza complexităţii
- implementarea algoritmului

Aceste etape nu sunt distincte în multe dintre cazuri, sau se ignoră, rezultatul constând în rezolvarea unei probleme pe calculator este ţelul final.

Reguli pentru realizarea unor algoritmi eficienţi

Dacă se vor concepe programele într-un mod cât mai general vor deveni adesea şi ceva mai uşor de scris, şi cu siguranţă foarte uşor de utilizat. Numai experienţa îi va convinge pe micii programatori de utilitatea unei discipline în actul programării. Deci vă prezint în continuare mai multe reguli importante care trebuie transmise elevilor:

1. Defineşte complet problema.

Această indicaţie, foarte importantă în activitatea de programare, pare fără sens pentru unii cititori. Dar nu se poate rezolva o problemă dacă nu se cunoaşte această problemă. Specificarea corectă şi completă a problemei nu este o sarcină trivială, ci una foarte importantă şi adeseori chiar dificilă. Programul trebuie să respecte această specificaţie, să fie construit având tot timpul în faţă această specificaţie, să i se demonstreze corectitudinea în raport cu această specificaţie, să fie testat şi validat ţinând seama de această specificaţie.

2. Gândeşte mai întâi, programează pe urmă.

Începând cu scrierea specificaţiilor problemei, trebuie pusă în prim plan gândirea. Este specificaţia problemei corectă? Între metodele de rezolvare posibile, care ar fi cea mai potrivită scopului urmărit? În paralel cu proiectarea algoritmului demonstrează corectitudinea lui. Verifică corectitudinea fiecărui pas înainte de a merge mai departe.

3. Nu folosi variabile neiniţializate.

Este vorba de prezenţa unei variabile într-o expresie fără ca în prealabil această variabilă să fi primit valoare. Este o eroare foarte frecventă a programatorilor începători (dar nu numai a lor). Destule compilatoare permit folosirea variabilelor neiniţializate, neverificând dacă o variabilă a fost iniţializată înaintea folosirii ei. Alte compilatoare iniţializează automat variabilele numerice cu valoarea zero. Cu toate acestea nu e bine să ne bazăm pe o asemenea iniţializare ci să atribuim singuri valorile iniţiale corespunzătoare variabilelor. Programul realizat trebuie să fie portabil, să nu se bazeze pe specificul unui anumit compilator.

4. Verifică valoarea variabilei imediat după obţinerea acesteia.

Dacă o variabilă întreagă trebuie să ia valori într-un subdomeniu $c_1..c_2$ verifică respectarea acestei proprietăţi. Orice

încălcare a ei indică o eroare care trebuie înlăturată. Valoarea variabilei poate fi calculată sau introdusă de utilizator. În primul caz, verificarea trebuie făcută după calcul, în al doilea caz se recomandă ca verificarea să urmeze imediat după citirea valorii respectivei variabile.

5. Cunoaşte şi foloseşte algoritmi clasici în rezolvarea problemelor.

Este vorba de algoritmi ca: Euclid pentru c.m.m.d.c., aflarea cifrelor unui număr, inversul unui număr, număr palindrom, număr prim, sortarea unui şir prin metoda bulelor, etc.

Folosirea unor componente existente (deci testate) scurtează perioada de realizare a acestora. Evident, dacă o parte din subalgoritmii necesari programului sunt deja scrişi şi verificaţi, viteza de lucru va creşte prin folosirea lor şi problemele sunt pe jumătate rezolvate.

6. Amână pe mai târziu detaliile nesemnificative.

Această regulă stabileşte priorităţile de realizare a componentelor unui program; în primul rând se acordă atenţie aspectelor esenţiale, începând cu modulul principal. În fiecare fază dă atenţie lucrurilor importante. De exemplu, este inutil să se piardă timp cu scrierea unor părţi de program pentru tipărirea rezultatelor şi

a constata ulterior că rezultatele nu sunt cele dorite, sau nu sunt corecte.

Nu uita însă că pentru beneficiar "**Detaliile nesemnificative sunt semnificative**". Beneficiarii ţin foarte mult la forma rezultatelor şi, adeseori, judecă programatorii după această formă. E păcat de munca depusă dacă tipărirea rezultatelor lasă o impresie proastă asupra beneficiarului.

7. Evită artificiile.

Prin folosirea artificiilor în programare, a prescurtărilor şi simplificărilor se pierde adesea din claritatea programului şi, mult mai grav, uneori se ajunge chiar la introducerea unor erori. În plus se poate pierde portabilitatea programului.

Există însă situaţii în care prin anumite artificii se câştigă eficienţă în execuţie sau se face economie de memorie. Dacă acest fapt este important atunci artificiile sunt binevenite, în caz contrar nu se recomandă folosirea lor.

8. Foloseşte constante simbolice.

Folosirea intensivă a constantelor simbolice este recomandată oriunde în textul sursă trebuie scris un număr (la declararea tablourilor, la precizarea limitelor de variaţie a unor variabile, etc.). Prin utilizarea acestor constante se măreşte gradul de

generalitate a textului scris, iar în situaţia în care valoarea unei constante trebuie schimbată, modificarea este mult mai uşoară (doar la locul definiţiei constantei) şi nu duce la erori. Ea implică numai definiţia constantei, nu modificarea valorii concrete în toate instrucţiunile programului.

9. Verifică corectitudinea algoritmului şi programului în fiecare etapă a elaborării lor.

Detectarea şi eliminarea unei erori imediat după comiterea ei duce la creşterea vitezei de realizare a produsului, evitându-se activităţi inutile de depanare. Se recomandă demonstrarea corectitudinii fiecărui algoritm folosit, întrucât erorile semnalate în timpul testării sunt adeseori greu de descoperit şi, câteodată, imposibil de eliminat altfel decât prin rescrierea modulului sau programului respectiv. Urmează testarea fiecărui subprogram imediat după ce a fost scris (codificat). Acest lucru se potriveşte codificării bottom-up şi sugerează o abordare sistematică a activităţii de codificare. Dacă pentru proiectare se pot folosi oricare dintre metodele indicate, în codificare (şi testarea aferentă codificării), abordarea de jos în sus este esenţială. Sugerăm ca această testare să se facă independent de programul în care se va folosi subprogramul testat. Este adevărat că activitatea de testare necesită un anumit timp, dar ea este utilă cel puţin din trei puncte de vedere:

- scoate în evidenţă erorile provocate de proiectarea algoritmului sau codificarea neadecvată a acestuia;

- facilitează detectarea erorilor, deoarece dimensiunea problemei este mai mică; în fapt nu se pierde timp cu scrierea unui program de test, ci se câştigă timp, deoarece la fiecare nivel de detaliere se vor folosi numai componente testate deja; ceea ce rămâne de testat la nivelul respectiv este gestiunea corectă a apelurilor respectivelor componente;

- obligă implementatorul să gândească încă o utilizare (cel puţin) a respectivului subprogram, independentă de cea pentru care a fost iniţial conceput.

10. Foloseşte denumiri sugestive pentru identificatorii utilizaţi în program.

Fiecare identificator (nume de variabilă, de tip de date, de constante, de subprograme) îşi are rolul şi semnificaţia lui într-un program. E bine ca denumirea să reflecte această semnificaţie, mărind astfel claritatea textului programului.

Unii programatori exagerează însă, folosind identificatori lungi, obţinuţi prin concatenarea mai multor cuvinte. E clar că denumirea aleasă redă semnificaţia variabilei, dar claritatea textului scade, lungimea programului creşte şi citirea lui devine greoaie.

11. Cunoaşte şi respectă semnificaţia fiecărei variabile.

Fiecare variabilă are o semnificaţie. În demonstrarea corectitudinii algoritmului această semnificaţie se reflectă, de cele mai multe ori, printr-un predicat invariant. O greşeală frecventă făcută de unii programatori constă în folosirea unei variabile în mai multe scopuri.

12. Foloseşte variabile auxiliare numai acolo unde este strict necesar.

Fiecare variabilă trebuie să aibă o semnificaţie proprie, iar în demonstrarea corectitudinii programului, acesteia i se ataşează un invariant, care trebuie verificat.

Folosirea necontrolată a mai multor variabile auxiliare, ruperea unor expresii chiar lungi în subexpresii cu diferite denumiri, pot duce la reducerea clarităţii programului.

13. Prin scriere redă cât mai fidel structura programului.

Importanţa indentării şi spaţierii pentru claritatea programului au fost arătate anterior. Fiecare programator trebuie să aibă propriile reguli de scriere, care să scoată cât mai bine în evidenţă structura programului şi funcţiile fiecărei părţi a acestuia.

14. Nu uita să testezi programul chiar dacă ai demonstrat corectitudinea lui.

Sunt cunoscute demonstraţii greşite pentru unele teoreme celebre din matematică. Şi o demonstraţie a corectitudinii unui program poate fi greşită. Dar, chiar dacă demonstrarea corectitudinii algoritmului este validă, programul poate conţine greşeli de codificare, de introducere (tastare) sau pot fi alte cauze care generează erori.

15. Nu recalcula limitele şi nu modifica variabila de ciclare în interiorul unei structuri repetitive dată prin propoziţia Pseudocod PENTRU.

O astfel de practică poate duce la erori greu de detectat şi încalcă regulile programării structurate. Atunci când este necesară schimbarea variabilei de ciclare sau a limitelor se recomandă folosirea uneia din structurile repetitive REPETĂ sau CÂTTIMP.

16. Nu ieşi forţat din corpul unei structuri repetitive redată prin propoziţia Pseudocod PENTRU.

Instrucţiunea Pseudocod PENTRU corespunde unui număr cunoscut de execuţii ale corpului ciclului. În situaţia când corpul conţine şi testarea condiţiei de continuare a ciclării, recomandăm a se folosi structurile REPETĂ sau CÂTTIMP şi nu PENTRU.

17. Elaborează documentaţia programului în paralel cu realizarea lui.

Aşa cum s-a arătat în mai multe locuri din acest material, pe durata de viaţă a unui program se iau mai multe decizii. E bine ca aceste decizii să rămână consemnate împreună cu rezultatul final al fiecărei faze din viaţa programului (specificarea problemei, proiectarea algoritmilor, programul propriu-zis, datele de test folosite). Vor rezulta documentaţii de analiză, proiectare, implementare şi exploatare. Primele trei sunt necesare la întreţinerea aplicaţiei, trebuind a fi actualizate ori de câte ori se produc modificări, iar ultima este necesară celor care exploatează aplicaţia. Pe lângă acestea, un program bun va trebui să posede şi o componentă de asistenţă on-line (funcţie *help*), care contribuie la asigurarea a ceea ce am numit *interfaţă prietenoasă*.

18. **Foloseşte comentariile**.

Este foarte greu să descifrăm un program lipsit de comentarii, chiar dacă este vorba de propriu; program scris în urmă cu câteva luni sau ani de zile. Orice program sau modul trebuie să fie însoţit de comentarii explicative dacă dorim să-l refolosim şi nu trebuie să scriem programe care să nu poată fi refolosite. Minimum de comentarii într-un modul trebuie să conţină specificarea acestui modul şi semnificaţia fiecărei variabile.

Etapele realizării unui program

Realizarea unui program pentru o anumită aplicaţie presupune implicarea mai multor etape. Aceste etape sunt independente de limbajul de programare utilizat şi implică existenţa câtorva restricţii cu privire la computerul utilizat.

Etapele realizării unui program sunt următoarele:

Studierea detaliată a cerinţelor aplicaţiei. Este foarte important ca cerinţele impuse de aplicaţie să fie foarte bine explicitate. Adică înainte de a trece la realizarea unui program pentru o anumită aplicaţie trebuie ca cea aplicaţie sa fie foarte bine analizată şi cerinţele pe care aceasta le impune trebuie să fie complete şi consistente. De exemplu o cerinţa de genul "scrie un program care sa rezolve ecuaţiile" este evident că este incompletă şi se impun întrebări de genul "ce tip de ecuaţii", "câte ecuaţii", "care este precizia", etc.

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

Traducerea algoritmului realizat la etapa anterioară într-un limbaj de programare evoluat corespunzător. Forma scrisă a acestui program este denumită program sursă (source program sau source code). În această etapă programul trebuie citit şi verificat

pentru a i se stabili corectitudinea. Aceasta se face prin introducerea unui set de valori şi verificarea dacă programul furnizează valorile corespunzătoare corecte. O dată verificat programul este introdus în computer prin intermediul unui Editor.

Compilarea programului în limbaj maşină. Astfel programul obţinut în limbaj maşină se numeşte object code. În această etapă compiler-ul poate determina erori de sintaxă ale programului. O eroare de sintaxă este o greşeală în gramatica limbajului. de exemplu limbajul Pascal necesită ca fiecare linie să se termine cu „;”. Dacă se uită plasarea ; atunci compiler-ul va semnala eroarea de sintaxă. Astfel se repetă compilarea până la eliminarea tuturor erorilor de sintaxă.

Programul obţinut în urma compilării, object code, este apoi corelat (linked) cu o serie de biblioteci de funcţii (function libraries) care sunt furnizate de sistem. Toate acestea se petrec cu ajutorul unui program numit linker iar apoi programul linked object code este încărcat în memoria computerului de către un program numit loader.

Rularea programului compilat, linked şi încărcat cu un set de date pentru testare. Astfel se vor pune în evidenţă erorile de logică ale programului. Erorile de logică sunt erori care sunt produse de metoda de rezolvare a problemei. Astfel deşi programul este scris corect din punct de vedere al sintaxei acesta

poate executa ceva ce este incorect în contextul aplicaţiei. Poate fi ceva simplu, de exemplu realizarea unei operaţii de scădere în loc de adunare. O formă particulară a erorilor de logică este apariţia erorilor de rulare (run-time error). O eroare de rulare va produce o oprire a programului în timpul execuţiei pentru că nu anumite instrucţiuni nu pot fi realizate. De exemplu o împărţire la zero sau încercarea de accesare a datelor dintr-un fişier inexistent. Astfel se impune ca în această etapă programul să fie reverificat şi apoi erorile să fie recorectate prin intermediul Editorului ceea ce impune ca etapele 3,4, şi 5 să fie repetate până la obţinerea rezultatelor satisfăcătoare.

Programul poate fi pus în execuţie pentru rezolvarea problemei pentru care a fost conceput. Este posibil ca pe parcursul execuţiei sale să se mai depisteze anumite erori de logică. Astfel se impune reformularea algoritmului şi reluarea etapelor de realizarea a programului.

1.2. INIŢIERE ÎN PROGRAMARE PRIN APLICAŢII SIMPLE, DAR ATRACTIVE

În general, copiii din ziua de azi iau legătura cu minunata lume a calculatoarelor prin intermediul jocurilor. Desigur, jocurile pe calculator ajută la formarea unor deprinderi de bază în utilizarea calculatoarelor. Dar există unii copii care vor să afle mai

multe despre calculatoare, despre cum se crează jocurile sau alte soft-uri, utilizarea unor aplicaţii fiind pentru ei plictisitoare. Adică, fără să-şi dea seama ei de fapt doresc să înveţe programare, aceasta ajutând la dezvoltarea gândirii critice, creative, comparative, analogice. Acest lucru nu este uşor, dar experienţa demonstrează că iniţierea în programare este benefică de la o vârstă cât mai fragedă. Copiii nu au teamă să "comande" calculatorului, să caute soluţii, mintea lor este într-o continuă căutare de nou şi ar fi bine ca totul să înceapă ca o joacă. Prin exemplele pe care le voi da vă propun să începem să-i iniţiem pe cei mici în programare prin crearea unor jocuri, la început simple, apoi din ce în ce mai complexe.

Toate problemele trebuie să fie formulate în aşa fel încât să le trezească interesul, să-i îndemne să gândească creativ. Prin rezolvarea acestora le vom oferi cunoştinţele teoretice de bază necesare realizării programelor. Sub îndrumarea profesorului, copiii vor face un pas uriaş, vor avea mai multă încredere în ei, vor dori să participe la concursuri de programare şi vor fi încurajaţi să rezolve cu ajutorul calculatorului probleme întâlnite la alte discipline (matematică, fizică, chimie etc.). Iată câteva exemple:

1. Conversaţie elev-calculator

Este o aplicaţie simplă în care vom utiliza instrucţiunile de citire-scriere şi cea de decizie pentru a simula o discuţie între elev şi calculator.

Elevii vor căuta să folosească fraze cât mai haioase pentru ca programul lor să fie apreciat de colegi. Pentru a-i motiva pe elevi, aceştia vor schimba locurile de la calculatoare în momentul în care vor rula programele.

```
program discutie;
var nume, raspuns:string;varsta:longint;
{var=zona de memorie;string=sirde
caractere;integer=numar intreg;}
begin{marcheaza inceputul oricarui program}
  writeln('Salut! Cum te numesti?');
  {write=scrie pe ecran. Tot ce este intre
  apostofuri apare pe  ecran, tot ce nu este
  intre apostrofuri se ia din memorie}
  readln(nume);{read=citeste de pe ecran si
  salveaza in memorie}
  write('Salut! ' ,nume,' Ce varsta ai?');
  readln(varsta);{ln=efectueaza trecerea la
  randul urmator}
  if (varsta<=0) then writeln('Esti prea tanar
  pentru a te crede!!! Chiar asa este????');
  {daca conditia este adevarata se executa
  instructiunile ce urmeaza dupa then}
  if (varsta>1) and (varsta<4) then
  writeln('Esti bebelus!!!!Nu poti scrie,
  Minti!!!' );
  if (varsta>=100) then writeln('Ti-au marit
  pensia ca ai trecut de 100 de ani????');
  if (varsta>100) or (varsta<=0) then
  writeln('Nu cred ca ai varsta asta!!!');
```

```
  if (varsta>=4) and (varsta<7) then
  writeln('Esti la gradinita? Ai jucarii
  frumoase?');
  readln(raspuns);
end.
```

2. Învaţă tabla înmulţirii

Algoritmul foloseşte funcţia random pentru a alege aleator două numere din intervalul 1-10 şi verifică rezultatul înmulţirii celor două numere. Operaţiile se repetă până la introducerea unui număr negativ.

Acest program va fi utilizat de elevii din clasele a III-a la orele de matematică. De aceea trebuie să conţină şi mesaje adecvate pentru deveni atractiv.

```
program verfic;
uses crt;
var nr1,nr2,rez,nrg,i:integer;
    nume:string;
begin
 randomize;nrg:=0;rez:=1;
 writeln('Cum te numesti?');
 readln(nume);
 while (rez>0)do
 {Intrebarile se opresc atunci cand elevul
introduce un numar negativ}
 begin
  nr1:=random(9)+1;
  nr2:=random(9)+1;
  write('  ',nr1,'*',nr2,'=');
  readln(rez);
```

```
  if (rez=nr1*nr2 ) then writeln('Ai raspuns
corect ',nume)
   else
    if(rez>0) then
               begin
                writeln('  Raspunsul corect era
',nr1*nr2 ,' ',nume);
                nrg:=nrg+1;
               end;
 end;
 writeln('Ai primit nota ',10-nrg,' deoarece ai
',nrg,' greseli');
 readln;
end.
```

3. Ghiceşte numărul

Este un joc care poate fi realizat cu elevii din clasa a V-a. Calculatorul alege un număr la întâmplare cuprins între 1 şi un număr maxim dat, elevul trebuie să îl ghicească primind indicaţii dacă numărul este prea mare sau prea mic. Calculatorul va afişa numărul de încercări realizate până la ghicirea numărului.

```
program ghiceste;
uses crt;
var
nrcalc,nrmeu,i,inc,incf,nr_max:integer;nume,nume
f:string;
begin
 inc:=0;
 clrscr;
 writeln('Care este numarul maxim la care se
poate "gandi" calculatorul?');
 readln(nr_max);
 randomize;
 nrcalc:=random(nr_max)+1;
```

```
 writeln('Cum te numesti ?');readln(nume);
 while (nrcalc<>nrmeu) do
 begin
    write('Nrtau=');readln(nrmeu);
     if nrcalc=nrmeu then writeln('Ai ghicit din
',inc,' incercari', nume);
    else begin
     inc:=inc+1;
     if nrmeu>nrcalc then writeln('Numarul este
prea mare!',nume)
                      else writeln('Numarul este
prea mic!',nume) ;
     writeln('Mai incearca');
   end;
 end;
 readln;
end.
```

4. Mişcare

Este o problemă care simulează mişcarea unui cuvânt, mesaj sau orice alt simbol, pe ecran în sensul acelor de ceasornic. Elevii vor înţelege că mişcarea se va realiza prin succesiunea rapidă a unor operaţii executate de calculator şi anume: poziţionare cursor, scrie mesaj, pauză, mută cursorul înapoi, şterge mesajul prin scrierea unor spaţii. Se va folosi tipul string, instrucţiunile de citire-scriere şi instrucţiunile gotoxy şi delay.

```
program misc;
uses crt;
var i,j,x,y,lg:integer;nume,spatiu:string;
begin
writeln('Cum te numesti?');readln(nume);
lg:=length(nume);spatiu:='';
```

```
for i:=1 to lg do
spatiu:=spatiu+' ';
clrscr;
for i:=1 to 75-lg do
begin
gotoxy(i,1);write(nume);delay(100);gotoxy(i,1);w
rite(spatiu);
end;
for i:=1 to 25 do
begin
gotoxy(75-
lg,i);write(nume);delay(100);gotoxy(75-
lg,i);write(spatiu);
end;
for i:=75-lg downto 1 do
begin
gotoxy(i,25);write(nume);delay(100);gotoxy(i,25)
;write(spatiu);
end;
for i:=25 downto 1 do
begin
gotoxy(1,i);write(nume);delay(100);gotoxy(1,i);w
rite(spatiu);
end;
readln;
end.
```

5. Alergătorul

În anii ‘90, se organizau concursuri de programare la Palatul Copiilor, unde participau elevi începând de la clasa a II-a. Atunci erau calculatoare HC85 şi se utiliza programul QBasic.

Îmi aduc aminte enunţul unei probleme dată la concurs, la clasa a II-a: *Un alergător parcurge conturul stadionului doborând*

la fiecare colţ un obstacol. Să se simuleze mişcarea alergătorului în jurul stadionului.

Enunţul problemei poate fi modificat astfel că alergătorul poate parcurge de mai multe ori conturul stadionului (în circuit), iar numărul de ture este stabilit la începutul probei.

Se vor folosi:

uses crt; - este vorba de biblioteca (unit) de funcţii si proceduri care facilitează lucrul in modul text in Pascal. Daca linia asta lipseşte nu veţi putea folosi o parte din funcţiile si procedurile de prin program.

Clrscr - (clear screen) şterge orice text de pe ecran şi readuce culorile de afişare la valorile predefinite.

GotoXY (coloana, linia) - setează poziţia cursorului de text. Determină locul de pornire pentru următoarea afişare.

TextColor - setează culoarea textului (în modul text). Definită în unitul CRT.

TextBackGround - setează culoarea fondului (în modul text). Definită în unitul CRT.

Indicaţii:

-Stadionul are formă dreptunghiulară şi va fi de fapt ecranul. În modul text, ecranul văzut de noi dispune de 25 de linii şi 80 de coloane (noi o sa folosim 25 de linii si 75 de coloane). Un caracter afişează la intersecţia dintre o linie şi o coloană

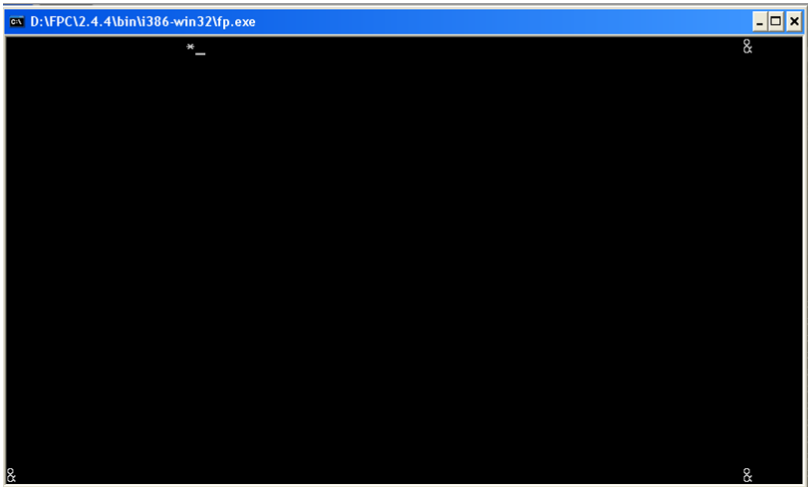
-Obstacolele vor fi reprezentate printr-un simbol, care va fi afişat în cele patru colţuri ale ecranului, de exemplu &

-Alergătorul va fi reprezentat tot printr-un caracter, de exemplu *

```
program alergare;
uses crt;
var i,j:integer;
    nume:string;
begin
 writeln('Cum te numesti?');readln(nume);
 clrscr;
 gotoxy(1,1);write('&');gotoxy(75,1);write('&');
 gotoxy(1,25);write('&');gotoxy(75,25);write('&'
);
 delay(500);
 for i:=1 to 75 do
 begin
  gotoxy(i,1);write('*');delay(50);gotoxy(i,1);w
rite(' ');
 end;
 for i:=1 to 25 do
 begin
  gotoxy(75,i);write('*');delay(50);gotoxy(75,i)
;write(' ');
 end;
 for i:=75 downto 1 do
 begin
  gotoxy(i,25);write('*');delay(50);gotoxy(i,25)
;write(' ');
 end;
 for i:=25 downto 1 do
 begin
    gotoxy(1,i);write('*');delay(50);gotoxy(1,i)
;write(' ');
 end;
 textcolor(red);textbackground(green);
```

```
 gotoxy(20,12);write('Felicitari ',nume,'!Ai
terminat o cursa!');
 readln;
end.
```

Enunţul problemei poate fi modificat astfel că alergătorul poate parcurge de mai multe ori conturul stadionului (în circuit), iar numărul de ture este stabilit la începutul probei. Se va folosi instrucţiunea FOR.



6. Piticul

Un pitic vrea sa urce o scară care are n trepte de înălţimi date ordonate crescător. Înălţimile treptelor sunt date în centimetri şi sunt valori întregi. Acolo unde diferenţa între două trepte consecutive este de 1 cm, piticul urcă fără dificultăţi. Unde

diferenţa este > de 1 cm, piticul trebuie să ia o pastilă care îi dă putere să sară pe treapta următoare. Cunoscându-se înălţimile treptelor, prima fiind 0, piticul vrea să afle care este numarul minim de pastile de care are nevoie pentru a urca scara şi de asemenea care este cea mai mare diferenţă între doua trepte consecutive.

```
program pitic;
var n,i1,i2,nrp,difmax,i:integer;
{n= numarul de trepte,i1=inaltimea primei
trepte, i1=inaltimea urmatoarei trepte,
nrp=numarul de pastile, difmax=diferenta maxima
dintre doua trepte consecutive, i=folosit pentru
parcurgerea scarilor}
begin
 writeln('numarul de trepte=');
 readln(n);{am salvat numarul de trepte in
 variabila n}
 nrp:=0;{la inceput piticul nu a luat nici o
 pastila}
 i1:=0;{inaltimea primei trepte primeste
 valoarea 0, pt. ca asa spune problema}
 {noi stam pe treapta i1 care la inceput are
 inaltimea 0}
 difmax:=0;{este un maxim deci primeste valoarea
 0 la inceput}
 {parcurgem toate cele n trepte cu ajutorul unui
 for}
 for i:=2 to n do {n a fost introdus mai sus de
 la tastatura}
 {Plecam de la 2 deoarece cunoastem inaltimea
 treptei 1=0(i1:=0)}
    begin
        writeln('inaltimea treptei ',i,'=');
```

```
        readln(i2);
        {daca diferenta dintre treapta i2 si
      treapta i1>1
        atunci piticul ia o pastila->crestem
      numarul de pastile cu 1}
        if(i2-i1>1) then nrp:=nrp+1;
        {nrp primeste valoarea(:=) vechea
      valoare a lui nrp +1}
        {daca diferenta dintre treapta i2 si
      treapta i1 >difmax atunci acea diferenta
      devine difmax}
        if (i2-i1>difmax) then difmax:=i2-i1;
        i1:=i2;{urcam o treapta}
    end;
 writeln('piticul a folosit ',nrp,' pastile, iar
 diferenta maxima dintre trepte este ', difmax);
 readln;
end.
{Observatie: numerele introduse de la tastatura
reprezinta inaltimile scarilor.
Fiecare inaltime a unei trepte se masoara
incepand de la pamant. De aceea inaltimile
introduse sunt numere crescatoare}
```

1.3. EXEMPLE DE PROBLEME REZOLVATE PENTRU GIMNAZIU

Dacă iniţierea în programare se face de la ciclul primar, atunci la clasele V-VIII se pot realiza programe cu grad de dificultate mai ridicat. Se vor introduce vectorii bidimensionali, fişierele text, procedurile şi funcţiile, codul ASCII etc. Orele se vor desfăşura în laboratorul de informatică, accentul se va pune pe partea practică, aplicativă, urmând ca elevii să modifice singuri programele îmbunătăţindu-le şi aducând o notă de originalitate proprie.

1. Roboţelul (clasa a VIII-a)

Folosind codul ASCII, construieşte un roboţel din caractere, pe care să-l deplasezi pe ecran în patru direcţii (sus, jos, stânga şi dreapta).

Roboţelul va fi construit folosind funcţia CHR care se aplică datelor de tip întreg, dar rezultatul este de tip Char. CHR returnează caracterul cu codul respectiv din codul ASCII. Deplasarea roboţelului se va face cu tastele W(sus), X(jos), S(stânga) şi D(dreapta).

```
program misc;
uses crt;
var i,j,x,y,lg,c,l:integer;tasta:char;

procedure scriu(c,l:integer);
begin
     gotoxy(c,l);write(chr(219));
```

```
     l:=l+1;c:=c-1;gotoxy(c,l);

write(chr(177),chr(177),chr(177));l:=l+1;gotoxy(
c,l);

write(chr(177),chr(177),chr(177));l:=l+1;gotoxy(
c,l);
     write(chr(186),'
',chr(186));l:=l+1;gotoxy(c,l);
     write(chr(186),' ',chr(186));l:=l-3;c:=c-
1;gotoxy(c,l);
     write(chr(201));l:=l+1;gotoxy(c,l);
write(chr(186));
     l:=l-
1;c:=c+4;gotoxy(c,l);write(chr(187));l:=l+1;goto
xy(c,l);
     write(chr(186));
end;
procedure sterg(c,l:integer);
begin
  gotoxy(c,l);write(' ');
  l:=l+1;c:=c-2;gotoxy(c,l);write('     ');
  l:=l+1;gotoxy(c,l); write('
');l:=l+1;gotoxy(c,l);write('    ');
  l:=l+1;gotoxy(c,l); write('     ');
  end;
begin
c:=20;l:=20;
clrscr;tasta:='q';scriu(c,l);
repeat
tasta:=readkey;
if (tasta='s') or (tasta='S') then{s=stanga}
begin
   sterg(c,l);
   if(c>=4)then c:=c-1;{protectia de a nu iesi
din ecran la stanga}
   scriu(c,l);
end;
```

```
if (tasta='d') or (tasta='D') then{d=dreapta}
begin
   sterg(c,l);
   if (c<=75) then c:=c+1;{protectia de a nu
iesi din ecran la dreapta}
   scriu(c,l);
end;
if (tasta='w') or (tasta='W') then {w=sus}
begin
   sterg(c,l);
   if(l>=2) then l:=l-1;    {protectia de a nu
iesi din ecran in sus}
   scriu(c,l);
end;
if (tasta='x') or (tasta='X') then {x=jos}
begin
   sterg(c,l);
   if(l<=20) then l:=l+1; {protectia de a nu
iesi din ecran in jos}
   scriu(c,l);
end;
until (tasta='q') or (tasta='Q');{q=iesire=esc}
readln;
end.
```

2. Moş Crăciun (clasa a VI-a)

Moş Crăciun având n saci de jucării în sanie s-a gândit să lase doi saci pentru o familie cu mai mulţi copii. Cunoscând câte jucării se află în fiecare sac, Moşul trebuie să aleagă sacii astfel încât numărul cadourilor rămase să fie cât mai mare şi să se împartă exact la cei n copii care îl mai aşteptă.

Exemplu n=6 În saci se găsesc numerele de jucării 8,3,12,9,10,7

Moşul lasă sacii 2 şi 5, fiecare din cei n copii vor primi câte 6 cadouri

Întrebări ajutătoare:

- Câţi saci are Moşul în sanie şi pe la câţi copii trebuie să mai treacă după aceea?
- Ce structură vom folosi pentru a memora numărul de jucării din fiecare sac?
- Ce variabile vom folosi?
- Cum vom introduce în memoria calculatorului numărul de saci şi câte jucării se află în fiecare sac?
- Ce modalităţi de rezolvare a problemei propuneţi?
- Cu ce problemă rezolvată anterior de noi se aseamănă?
- Cum putem alcătui toate perechile posibile de câte 2 saci?
- Ce condiţii punem acestor perechi pentru a finaliza problema?

```
program Mos_Craciun;
Var
n,i,j,s1,s2,suma,max:integer;v:array[1..100]of
integer;
begin
write('Numarul                                    de
saci=');readln(n);suma:=0;max:=0;
for i:=1 to n do
begin
write('Cate jucarii se afla in sacul',i,'?');
readln(v[i]);suma:=suma+v[i];end;
for i:=1 to n-1 do
  for j:=i+1 to n do
      if  (suma-v[i]-v[j]>max)  and  ((suma-v[i]-
v[j]) mod n =0) then
              begin   s1:=i;s2:=j;max:=suma-v[i]-
v[j];end;
writeln('Au fost lasati sacii ',s1,',',s2);
writeln('Fiecare  copil  a  primit  ',(suma-v[s1]-
v[s2]) div n,'jucarii');
readln;
end.
```

Pentru consolidarea şi fixarea noilor cunoştinţe se propune spre rezolvare aceeaşi problemă, pentru cazul în care Moşul lasă 3 saci de jucării.

3. Meteoriţi (clasa a VIII-a)

Suntem programatori pe o staţie spaţială care urmează să suporte o cădere de meteoriţi. Staţia dispune de senzori ce ne pot furniza într-un fişier text, distanţa la care se află fiecare meteorit şi viteza cu care se apropie. Vom identifica meteoriţii cu numărul de apariţie din fişierul text. Pentru a se apăra, staţia are în dotare o armă cu laser ce poate distruge cu precizie şi instantaneu fiecare meteorit. Pentru a evita o catastrofă, avem nevoie de un program care să-i "comunice" armei ordinea de distrugere meteoriţilor. În mod normal primul meteorit care va fi distrus va fi cel care va lovi primul staţia şi aşa mai departe. Se dă fişierul de intrare meteo.in ce are pe prima linie numărul n al meteoriţilor, iar pe următoarele n linii câte 2 pe linie, separate printr-un spaţiu, distanţa, respectiv viteza fiecăruia. Se cere să se scrie în fişierul de ieşire meteo.out, ordinea de distrugere a acestora.

Exemplu:

meteo.in	meteo.out
5	3 1 4 5 2
24 8	
240 10	
6 3	
10 2	
18 2	

Întrebări ajutătoare:

- Cum obţinem timpul cunoscând distanţa şi viteza?
- Cum structurăm informaţia?
- Cum salvăm informaţia din fişier în vector?
- Ce tip de date trebuie să conţină vectorul?

- Cum sortăm vectorul?
- Cum scriem informaţia în fişierul de ieşire?

```
program meteor;
var
n,i,j,ap:integer;at,dis,vit:real;t:array[1..
100]of real;
    p:array[1..100]of
integer;sortat:boolean;f,g:text;
begin
assign(f,'meteo.in');reset(f);readln(f,n);
for i:=1 to n do
 begin
readln(f,dis,vit);p[i]:=i;t[i]:=dis/vit;end;
 close(f);
 repeat
 sortat:=true;
 for i:=1 to n-1 do
   if t[i]>t[i+1] then begin
          at:=t[i];t[i]:=t[i+1];t[i+1]:=at;

ap:=p[i];p[i]:=p[i+1];p[i+1]:=ap;sortat:=fal
se;
                        end;
 until sortat=true;
 assign(g,'meteo.out');rewrite(g);
 for i:= 1 to n do write(g,p[i],' ');
 close(g);
 end.
```

4. Scooby-Doo (clasa a V-a)

Pentru a spăla maşina la râu, Shagy a pregătit 3 vase pentru apă de capacităţi a,b,c litri. Scooby Doo îl ajută şi îi cară apă cu un vas de capacitate d (d>a,d>b,d>c, vărsând apa pe

rând în fiecare dintre ele. De fiecare dată se formează o mică baltă. Determinaţi cantitatea de apă risipită după nrd drumuri.

Exemplu: a=3 b=4 c=5 d=6 nrd=5 S-au risipit 11L de apă.

Întrebări ajutătoare:

- De ce se formează o baltă?
- Cum determinam câtă apă s-a risipit pentru exemplul de mai sus?
- Cum determinăm în funcţie de numărul drumului câtă apă se risipeşte şi câte cazuri avem?
- Ce variabile vom folosi?
- Care sunt datele de intrare cunoscute şi cum le introducem în calculator?
- Ce instrucţiune vom folosi pentru a „parcurge” toate drumurile?
- Ce funcţie vom folosi pentru a determina câtă apă se risipeşte la fiecare drum?
- Cum vom calcula cantitatea de apă risipită la fiecare drum?

```
program Scooby_Doo;
var a,b,c,d,i,apr,nrd:integer;
begin
  write('a,b,c,d=');readln(a,b,c,d);
  write('nrd=');readln(nrd);
  apr:=0;
  for i:=1 to nrd do
    begin
```

```
      if i mod 3 =1 then apr:=apr+(d-a);
      if i mod 3 =2 then apr:=apr+(d-b);
      if i mod 3 =0 then apr:=apr+(d-c);
    end;
    writeln('S-au risipit ',apr,' litri');
    readln;
end.
```

Pentru consolidarea şi fixarea noilor cunoştinţe se propune rezolvarea acestei probleme în cazul în care Shagy are 5 vase a,b,c,d,e, iar Scooby-Doo cară apă cu un vas de capacitate f>a,b,c,d şi f<e.

5. Panglica (clasa a VII-a)

*Gigel are o panglică alcătuită din benzi de 1 cm lăţime, colorate în diverse culori. Panglica are **N** benzi colorate cu **C** culori, pe care le vom numerota de la **1** la **C**. Gigel vrea ca la ambele capete ale panglicii să aibă aceeaşi culoare, dar cum nu poate schimba culorile benzilor, singura posibilitate rămâne tăierea unor bucăţi de la capete.*

Cerinţă

Scrieţi un program care să determine modul de tăiere a panglicii astfel încât la cele două capete să fie benzi de aceeaşi culoare, iar lungimea panglicii obţinute să fie maximă.

Date de intrare

Fişierul de intrare **PANGLICA.IN** conţine:

– pe prima linie numerele naturale **N** şi **C** separate printr-un spaţiu;
– pe următoarele **N** linii descrierea panglicii: pe fiecare linie un număr natural de la **1** la **C**, reprezentând în ordine culorile fâşiilor ce alcătuiesc panglica.

Date de ieşire

Fişierul de ieşire **PANGLICA.OUT** va conţine următoarele 4 numere:

– pe prima linie numărul de fâşii rămase;
– pe linia a doua numărul culorii care se află la capete;
– pe linia a treia câte fâşii trebuie tăiate de la începutul panglicii iniţiale;
– pe linia a patra câte fâşii trebuie tăiate de la sfârşitul panglicii iniţiale.

Restricţii şi precizări

- **2≤N≤10000**
- **1≤C≤200**
- Dacă există mai multe soluţii alegeţi pe cea în care se taie cât mai puţin din partea de început a panglicii.

panglica.in	panglica.out	panglica.in	panglica.out
6 3	**4**	**5 2**	**4**
1	**2**	**1**	**2**
2	**1**	**2**	**1**
1	**1**	**1**	**0**
3		**2**	
2		**2**	
3			

Întrebări ajutătoare:

- Ce structură de date vom folosi pentru a memora componentele panglicii?
- Cum salvăm informaţia din fişier în variabile?
- Cum salvăm informaţia din fişier în vector?
- Ce tip de date trebuie să conţină vectorul?
- Cum prelucram informaţia din vector?
- Ce date trebuie reţinute după prelucrarea informaţiei?
- Cum scriem informaţia în fişierul de ieşire?

```
program panglica;
var p:array[1..10000] of byte; f,g:text;
   c1,c2,c1m,c2m,l,lmax,n,c:integer;
begin
  lmax:=0;{maximul porneste de la valoarea 0}
  assign(f,'panglica.in');reset(f);{Am deschis
fisierul pentru citirea datelor}
  readln(f,n,c);{citim din fisier pe n si c}
```

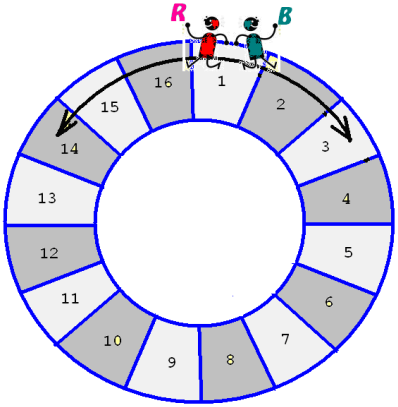
```
  for c1:=1 to n do
    readln(f,p[c1]);{citim informatia din fisier
direct in vector}
  close(f);{Inchidem fisierul deoarece am
terminat citirea datelor}
  for c1:=1 to n-1 do {parcurgem vectorul de la
ambele capete}
   for c2:=n downto c1+1 do
     if (p[c1]=p[c2]) then {daca capetele au
aceeasi culoare}
        if (c2-c1+1>lmax) then{daca lungimea
panglicii este> decat}
           begin c1m:=c1; {lungimea maxima,
retin acele capete}
                 c2m:=c2;
                 lmax:=c2-c1+1;
           end;
assign(g,'panglica.out');rewrite(g);{deschidem
fisierul pentru citirea datelor}
   writeln(g,lmax);{Scriem in fisier numarul de
fasii ramase
   sau lungimea maxima din panglica care acre
aceeasi culoare la ambele capete}
   writeln(g,p[c1m]);{scriem in fisier culoarea
capetelor}
   writeln(g,c1m-1);
   writeln(g,n-c2m);close(g);end.
```

1.4. PROBLEME REZOLVATE – OLIMPIADA DE INFORMATICĂ ONI-GIM ŞI OJI-GIM

Problema 1 – Joc *(ONI Gim Piatra-Neamţ, 15-22 aprilie 2011, clasa a VI-a)*

Rareş şi Bogdan vor să facă mişcare în aer liber aşa că s-au gândit la un nou joc.

Pe terenul de fotbal, ei au desenat o pistă circulară şi au împărţit-o în **n** sectoare egale, ca în desenul alăturat (unde **n=16**). Ei au etichetat cele **n** sectoare cu numerele distincte de la **1** la **n**, în sensul acelor de ceasornic.



Au stabilit ca jocul să se desfăşoare astfel:

- Se vor aşeza amândoi în sectorul numerotat cu **1**, spate în spate, astfel încât Bogdan se va deplasa în sensul acelor de ceasornic, iar Rareş în sens contrar.

Prin sărituri executate simultan în anumite sectoare, copiii se vor deplasa pe pistă în sensuri contrare şi vor executa un număr egal de sărituri.

- O săritură a lui Bogdan are ca efect deplasarea acestuia din sectorul curent, în sensul acelor de ceasornic, avansând cu **x** sectoare pe pistă. De exemplu, dacă **n=16** şi **x=2** atunci, pornind din sectorul **1**, Bogdan se va deplasa sărind succesiv, în această ordine, în sectoarele etichetate cu: **3,5,7,9,...**
- O săritură a lui Rareş are ca efect deplasarea acestuia din sectorul curent, în sens contrar acelor de ceasornic, avansând cu **y** sectoare pe pistă. De exemplu, dacă **n=16** şi **y=3** atunci, pornind din sectorul **1**, Rareş se va deplasa sărind succesiv, în această ordine, în sectoarele: **14,11,8,5,...**
- Jocul se termină când cei doi copii ajung în urma săriturilor într-un acelaşi sector de pe pistă sau dacă unul dintre cei doi copii ajunge pentru a doua oară într-un acelaşi sector.

Cerinţă

Scrieţi un program care să citească cele trei numere naturale nenule **n**, **x** şi **y**, şi apoi să determine:

a) numărul **t** al sectoarelor de pe pistă prin care **nu trece** niciunul dintre cei doi copii în urma săriturilor executate până la terminarea jocului;

b) numărul **s** de sărituri executate de fiecare copil până la terminarea jocului;

c) etichetele **b** şi **r** ale sectoarelor în care ajung cei doi copii la terminarea jocului (Bogdan ajunge la finalul jocului în sectorul cu eticheta **b**, iar Rareş în cel cu eticheta **r**).

Date de intrare

Fişierul de intrare **joc.in** conţine pe prima linie trei numere naturale **n**, **x** şi **y**, separate prin câte un spaţiu, cu semnificaţia din enunţ.

Date de ieşire

Fişierul de ieşire **joc.out** va conţine pe prima linie cele patru numere naturale determinate de program: **t**, **s**, **b** şi **r**, separate prin câte un spaţiu, în această ordine, cu semnificaţia din enunţ.

Restricţii şi precizări

- **16 ≤ n ≤ 40000; 1 ≤ x < n; 1 ≤ y < n**
- pentru rezolvarea corectă a primei cerinţe se acordă **20%** din punctaj, pentru rezolvarea corectă a celei de a doua cerinţe **40%** din punctaj şi pentru rezolvarea corectă a celei de a treia cerinţe **40%** din punctaj (**20%** pentru determinarea corectă a valorii **b**, respectiv **20%** pentru determinarea corectă a valorii **r**).

Exemple

<table>
<tr><th></th><th>joc.in</th><th>joc.out</th><th>Explicaţii</th></tr>
<tr><td>1)</td><td>16 2 3</td><td>4 8 1 9</td><td>Cei doi copii, executând simultan sărituri, trec până la terminarea jocului, prin sectoarele:
<table>
<tr><td>Bogdan</td><td>1</td><td>3</td><td>5</td><td>7</td><td>9</td><td>11</td><td>13</td><td>15</td><td>1</td></tr>
<tr><td>Rareş</td><td>1</td><td>14</td><td>11</td><td>8</td><td>5</td><td>2</td><td>15</td><td>12</td><td>9</td></tr>
</table>
Jocul se termină după s=8 sărituri deoarece Bogdan ajunge din nou în sectorul cu eticheta b=1. La finalul jocului Rareş se află în sectorul cu eticheta r=9. Cei doi copii nu au trecut prin t=4 sectoare ale căror etichete sunt: 4,6,10,16.</td></tr>
<tr><td>2)</td><td>16 6 2</td><td>12 2 13 13</td><td>Cei doi copii, executând simultan sărituri, trec până la terminarea jocului, prin sectoarele:
<table>
<tr><td>Bogdan</td><td>1</td><td>7</td><td>13</td></tr>
<tr><td>Rareş</td><td>1</td><td>15</td><td>13</td></tr>
</table>
Jocul se termină după s=2 sărituri deoarece Bogdan şi Rareş ajung amândoi în sectorul cu eticheta b=r=13. Cei doi copii nu au trecut prin t=12 sectoare ale căror etichete sunt: 2,3,4,5,6,8,9,10,11,12,14,16.</td></tr>
</table>

Timp maxim de executare/test: **1** secundă

Rezolvare:

```
program joc;
var
n,i,j,x,y:word;s,cb,cr:longint;f,g:text;r,b:long
int;
    v:array[1..40000]of char;tr:boolean;
 begin
   assign(f,'joc.in');reset(f);
   readln(f,n,x,y);
   close(f);
   b:=1;r:=n+1;
   b:=(b+x)mod n;
```

```
   if b=0 then b:=n;
   r:=r-y;
   v[r]:='r';
   v[b]:='b';
   tr:=false;s:=1;v[1]:='v';
   if r=b then begin
                 tr:=true;
                 v[b]:='v';
               end;
   repeat
      inc(s);
      b:=(b+x)mod n;
      if b=0 then b:=n;
      r:=r-y;
      if r<0 then inc(r,n);
      if r=b then begin
                    tr:=true;
                    v[b]:='v';
                  end;
      if tr=false then begin
      if (v[b]='b') or (v[b]='v') then tr:=true
                                  else if
v[b]='r' then v[b]:='v'

else v[b]:='b';
      if (v[r]='r') or (v[r]='v') then tr:=true
                                  else if
v[r]='b' then v[r]:='v'

else v[r]:='r';
                       end;
   until tr=true;
    j:=0;
   for i:=1 to n do
    if (v[i]<>'b')and(v[i]<>'r')and(v[i]<>'v')
then inc(j);
   assign(g,'joc.out');rewrite(g);
   writeln(g,j,' ',s,' ',b,' ',r);
```

```
  close(g);
end.
```

Probleme date la ONI Gim Cluj Napoca, 10-16 aprilie 2007,

clasa a VI-a

Problema 1 - ceas

Andrei a cumpărat un ceas de perete care are marcate orele unei zile cu ajutorul numerelor de la **1** la **12,** iar minutele sunt marcate cu puncte. Ceasul are două indicatoare. Primul indicator arată ora şi îşi schimbă poziţia din oră în oră. Al doilea indică minutul şi îşi schimbă poziţia din minut în minut. De exemplu, dacă este ora **10** şi **11** minute, indicatorul de oră este poziţionat pe numărul **10** marcat pe ceas, iar cel de minut este poziţionat pe punctul corespunzător celui de-al **11**-lea minut (ca în imaginea alăturată). După un minut, indicatorul de oră va fi poziţionat tot pe numărul **10,** iar cel de minut se va deplasa cu un punct, pentru a indica minutul **12**.
După câteva zile de funcţionare, Andrei observă că ceasul nu funcţionează corect, deoarece, de fiecare dată când cele două indicatoare se suprapun, ceasul stă în loc **5** minute (cele două indicatoare rămân suprapuse **5** minute).
Cunoscând ora şi minutul la care Andrei a fixat corect ceasul, determinaţi ce oră indică acesta după un anumit timp (exprimat în ore şi minute).

Cerinţă
Scrieţi programul care citeşte de la tastatură ora şi minutul la care e fixat ceasul, şi afişează pe ecran ora şi minutul indicate de ceasul lui Andrei, după un anumit număr de ore şi minute.

Date de intrare

Se citesc de la tastatură, de pe aceeaşi linie a ecranului, în această ordine, separate printr-un spaţiu, patru numere **h1 m1 h2 m2,** unde **h1** şi **m1** reprezintă ora şi minutul la care e fixat ceasul, **h2** şi **m2** reprezintă numărul de ore şi numărul de minute care au trecut de la fixarea acestuia.

Date de ieşire

Se vor afişa pe ecran, pe un singur rând, în această ordine, separate printr-un spaţiu, două numere **h3** şi **m3** ce vor reprezenta ora şi minutul indicate de ceas.

Restricţii şi precizări

- indicatorul de oră nu are poziţii intermediare, va fi întotdeauna poziţionat pe unul din numerele naturale din intervalul **[1,12]**

1 ≤ h1,h3 ≤ 12

0 ≤ h2 ≤ 1000

0 ≤ m1,m2,m3 ≤ 59

Exemplu:

Intrare	*Ieşire*	*Explicaţii*
2 30 1 10	**3 35**	Ceasul este fixat la ora **2** şi **30** de minute. După **30** de minute ceasul va indica ora **3** şi **0** minute. După alte **15** minute, indicatorul de oră şi cel de minut se vor suprapune, deoarece va fi ora **3** şi **15**. În această poziţie, cele două indicatoare mai rămân încă **5** minute(deoarece ceasul întârzie **5** minute). După alte **20** de minute ceasul va indica ora **3** şi **35** de minute.
3 7 2 19	**5 16**	Ceasul este fixat la ora **3** şi **7** de minute. După **2** ore şi **19** minute ceasul va indica ora **5** şi **16** minute.

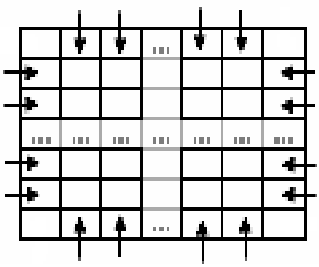
Timp maxim de execuţie/test: 1 secundă.

Rezolvare:

```
program ceas;
var h1,m1,h3,m3:integer;{h1,m1=ora de pornire a
ceasului}
    ms,mt,h2,m2:longint;{h3,m3=ora curenta pe
care o indica ceasul }
begin  {h2,m2=orele si minutele reale care trec
pana cand se verifica ceasul}
write('h1,m1=');readln(h1,m1);
write('h2,m2=');readln(h2,m2);
mt:=h2*60;{transform orele si minutele reale  in
minute}
mt:=mt+m2;
ms:=0;{ms=minutele reale scurse-> voi creste pe
ms pana cand ms>=mt}
h3:=h1;m3:=m1;{cand ms=0, ceasul arata ora
initiala}
repeat
   if m3=60 then begin{daca minutul este 60}
                  m3:=0; {, atunci minutul
devine 0}
                  h3:=h3+1;{iar ora pe care o
indica ceasul}
                  end;          {creste cu o
unitate}
   if h3=13 then {daca ora pe care o indica
ceasul este 13,}
               h3:=1;{ atunci ea devine 1}
   if m3=(h3*5 mod 60) then
    {daca limbile se suprapun} ms:=ms+5;{stau 5
minute}
   {adica ms se scurge, fara a indica nimic pe
ceas}
   if ms< mt then begin {daca minutele scurse
sunt mai putine decat}
                        ms:=ms+1;{mt=minutele
care trebuie sa treaca}
                        m3:=m3+1;end;
```

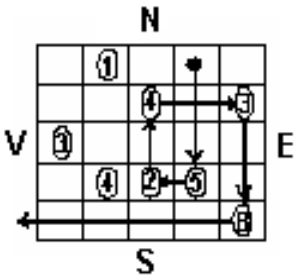
```
until ms>=mt;
writeln(h3,':',m3);
readln;
end.
```

Problema 2 – ouă



Pe o pajişte pătrată formată din **LxL** parcele pătrate cu latura de 1 metru au fost ascunse ouă. Unele ouă sunt mai valoroase decât altele. **N** iepuraşi se află în parcele situate pe marginea pajiştii şi participă la concursul „Coşul meu este mai valoros”. Iniţial nu există mai mulţi iepuraşi în aceeaşi parcelă. Ei poartă tricouri cu numere distincte de la **1** la **N**. Pentru că au foarte mult antrenament iepuraşii respectă cu stricteţe următoarele reguli: **Figura 1**

1) toţi iepuraşii încep căutarea ouălor simultan, pornind cu parcela în care se află;



2) direcţiile iniţiale de deplasare a iepuraşilor (reprezentate în figura 1) sunt următoarele: cei de pe latura nordică se deplasează către sud, cei de pe latura vestică se deplasează către est, cei de pe latura sudică către nord iar cei de pe latura estică către vest;

3) în fiecare parcelă un iepuraş stă un minut, timp în care caută şi eventual culege un ou şi sare în parcela indicată de direcţia sa de deplasare;

4) **culegerea** unui ou determină schimbarea direcţiei de deplasare;

Figura 2

5) schimbarea direcţiei de deplasare se face astfel: dacă iepuraşul vine de la **Nord** atunci el va pleca spre **Vest** , dacă vine de la **Sud**

va pleca spre **Est**, dacă vine de la **Est** va pleca spre **Nord** iar dacă vine de la **Vest** va pleca spre **Sud**. Un exemplu de deplasare este reprezentat în figura 2 pentru un iepuraş care pleacă din parcela (1,4);
6) dacă doi sau mai mulţi iepuraşi ajung simultan la acelaşi ou atunci acesta va fi cules de cel care are cel mai mic număr pe tricou.

Concursul se încheie atunci când nu mai există iepuraşi pe pajişte.

Cerinţă

Scrieţi un program care să afişeze numărul total de ouă culese de cei **N** iepuraşi, cel mai valoros coş (valoarea acestuia) precum şi durata (în minute) a concursului.

Date de intrare

Fişierul de intrare **oua.in** conţine:
Pe prima linie **2** numere naturale, **L** şi **P** separate printr-un spaţiu, **L** reprezentând numărul liniilor şi al coloanelor tabloului utilizat în reprezentarea parcelei iar **P** numărul ouălor de pe pajişte.
Pe fiecare dintre următoarele **P** linii se află câte **3** numere naturale, separate prin câte un spaţiu, reprezentând coordonatele parcelei **(i,j)** pe care se găseşte un ou **(**1<=i<=L;1<=j<=L **)** şi valoarea **(v)**acestuia.
Pe linia următoare se află numărul **N** al iepuraşilor, iar pe fiecare dintre următoarele **N** linii se găsesc perechi de numere naturale reprezentând coordonatele parcelelor în care se găsesc iniţial iepuraşii, **în ordinea numerelor de pe tricou**.

Date de ieşire

Fişierul de ieşire **oua.out** va conţine:
Pe prima linie se găsesc **3** numere naturale separate prin câte un spaţiu, numere ce reprezintă numărul total de ouă culese de cei **N** iepuraşi, cel mai valoros coş (valoarea acestuia) precum şi durata (în minute) a concursului.

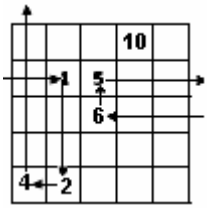
Restricţii şi precizări

2<=L<=50; 0<=N<=100; 0<=P<=2500; 1<=v<=30;

Un iepuraş iese de pe pajişte dacă ajunge în afara matricei.

La un moment dat pot exista mai mulţi iepuraşi în aceeaşi parcelă, cu excepţia primului minut.

Exemplu

oua.in	oua.out	*Explicaţie*
5 6 1 4 10 2 2 1 2 3 5 3 3 6 5 1 4 5 2 2 2 2 1 3 5	5 11 10	 Primul iepuraş stă **10** minute pe pajişte, culege **3** ouă, cu valoare totală 7 şi parcurge traseul (2,1)-(2,2)-(3,2)-(4,2)-(5,2)-(5,1)-(4,1)-(3,1)-(2,1)-(1,1)→ Al doilea iepuraş stă 6 minute pe pajişte, culege **2** ouă, cu valoare totală **11** şi parcurge traseul (3,5)-(3,4)-(3,3)-(2,3)-(2,4)-(2,5)→

Timp maxim de executare/test: 1 secundă.

Rezolvare:

```
program oua;
type r=record
l,c,v,d:integer;
end;
var
lt,nro,nri,v,i,j,l,c:integer;co:array[1..50,1..5
0]of integer;
pi:array[1..100]of r;f,g:text;gata:boolean;
begin
  assign(f,'oua.in');reset(f);readln(f,lt,nro);
  for i:=1 to lt do
    for j:=1 to lt do
      co[l,c]:=0;
  for i:=1 to nro do begin
readln(f,l,c,v);co[l,c]:=v;end;
```

```
  readln(f,nri);
  for i:=1 to nri do
  begin
    readln(f,pi[i].l,pi[i].c);
    pi[i].v:=0;
    if pi[i].l=1 then pi[i].d:=90;
    if pi[i].l=lt then pi[i].d:=270;
    if pi[i].c=1 then pi[i].d:=180;
    if pi[i].c=lt then pi[i].d:=0;
  end;close(f);
  repeat
    gata:=true;
    for i:=1 to nri do
     begin
        if pi[i].d=0 then pi[i].c:=pi[i].c-1;
        if pi[i].c<0 then pi[i].d:=-1
            else
                 if co[pi[i].l,pi[i].c]>0 then
begin
                              pi[i].v:=pi[i].v+
co[pi[i].l,pi[i].c];

co[pi[i].l,pi[i].c]:=0;
                              pi[i].d:=(pi[i].d
+90) mod 360;
                                        end;
     end;

    for i:=1 to nri do
        if pi[i].d>=0 then gata:=false;
  until gata;
end.
```

Problema 3 – turn

Ionică are o cutie plină cu cuburi. Pe fiecare cub este scrisă o cifră. La un moment dat Ionică aşează **n** cuburi unul peste altul formând un turn. Tatăl lui Ionică văzând turnul îi spune o cifră **k**

şi o modalitate de scoatere a unor cuburi din turn. Mai precis Ionică trebuie să scoată cuburi pentru a obţine un turn cu **înălţimea minimă** după următoarele reguli:

- la un moment dat se poate scoate o grupă cu cel puţin **k** cuburi alăturate care au pe ele scrise aceeaşi cifră;
- grupele se elimină începând de la baza turnului, de fiecare dată începând cu prima grupă ce respectă condiţia anterioară.

Cerinţă
Să se scrie un program care să determine turnul final, după eliminarea tuturor grupelor de cuburi conform modalităţii precizate de tatăl lui Ionică.

Date de intrare
Fişierul de intrare **`turn.in`** are pe prima linie numerele naturale **n** şi **k** separate printr-un spaţiu, iar pe linia următoare cifrele scrise pe cuburi în ordine de la baza cubului spre vârf, separate între ele prin câte un spaţiu.

Date de ieşire
Fişierul de ieşire **`turn.out`** va conţine pe prima linie numărul de cuburi din turnul cerut, iar a doua linie cifrele de pe cuburile turnului (în ordine de la baza turnului până la vârful său) cu un spaţiu între spaţiu între ele.

Restricţii şi precizări
- $2 \leq \mathbf{n} \leq 49000$
- $2 \leq \mathbf{k} \leq 9$
- pentru toate testele turnul rezultat are cel puţin un cub

Exemple

turn.in	**turn.out**	***Explicaţie***
20 3 1 0 2 2 2 0 0 0 7 7 5 5 5 5 5 7 7 7 3 9	3 1 3 9	Evoluţia turnului este următoarea: După eliminarea grupului de cuburi cu cifrele **2 2 2** se obţine: 1 0 0 0 0 7 7 5 5 5 5 5 7 7 7 3 9 După eliminarea grupului de cuburi cu cifrele **0 0 0 0** se obţine: 1 7 7 5 5 5 5 5 7 7 7 3 9 După eliminarea grupului de cuburi cu cifrele **5 5 5 5 5** se obţine: 1 7 7 7 7 7 3 9 După eliminarea grupului de cuburi cu cifrele **7 7 7 7 7** se obţine: 1 3 9
turn.in	**turn.out**	***Explicaţie***
21 4 1 2 2 2 2 1 1 3 3 3 3 3 1 1 4 4 4 4 4 1 1	2 1 1	Evoluţia turnului este următoarea: După eliminarea grupului de cuburi cu cifrele **2 2 2 2** se obţine: 1 1 1 3 3 3 3 3 1 1 4 4 4 4 4 1 1 După eliminarea grupului de cuburi cu cifrele **3 3 3 3 3** se obţine: 1 1 1 1 1 4 4 4 4 4 1 1 După eliminarea grupului

turn.in	turn.out	*Explicaţie*
		de cuburi cu cifrele **1 1 1 1 1** se obţine: 4 4 4 4 4 1 1 După eliminarea grupului de cuburi cu cifrele **4 4 4 4 4** se obţine: 1 1

Timp maxim de executare/test: 2 secunde

Rezolvare:

```
program turnul;
var
n,i,j,h,nr_ex,t:longint;k,cap:byte;am_extras:boo
lean;
    v:array[1..49000] of byte;f,g:text;
begin{cu i,j,t parcurgem vectorii }
 assign(f,'turn1.in');reset(f);readln(f,n,k);
 for i:=1 to n do   {k si cap sunt cifre de la 0
la 9 deci pot fi de tip byte}
   read(f,v[i]);   {tipul byte poate retine
valori de la 0 la 255 }
 h:=n;am_extras:=true;{si ocupa memorie
putina.Dupa extragere spatiul}
while am_extras=true do {il notam cu 10 pt. ca
nici o cifra nu poate lua val. 10}
begin
 am_extras:=false;
 for i:=1 to h-1 do
   begin
     if (v[i]=v[i+1]) then
         begin cap:=v[i];{daca am 2 val
alaturate capturez valoarea}
               j:=i;  {merg cat timp am ac.
valoare capturata}
               while (cap=v[j]) and (i<=h) do
j:=j+1;
```

```
               if j-i>=k then {daca lungimea
sirului de val egale >=k}
                  begin  {atunci inlocuiesc
sirul cu 0}
                  for t:=i to j-1 do begin
v[t]:=10;nr_ex:=nr_ex+1;end;
                     am_extras:=true;i:=h-1;{ies
fortat din for}
                  end;
          end;
     end;{daca scot nr_ex cuburi, celelalte
coboara mai jos cu nr_ex pozitii astfel}
  for t:=1 to h-nr_ex do
    if v[t]=10 then
      begin j:=t;
            while v[j]=10 do j:=j+1;
            v[t]:=v[j];v[j]:=10;
       end;
   h:=h-nr_ex;{micsorez inaltimea turnului
}i:=1;nr_ex:=0;
end;{Acum scriu in fisierul de iesire
g}Assign(g,'turn.out');rewrite(g);
writeln(g,h);{scriu in fis}writeln(h);{scriu pe
ecran}
for i:=1 to h do
begin write(g,v[i],' ');write(v[i],'
');end;close(f);close(g);
end.
```

Probleme date la OJI Gim, 19 martie 2006, clasa a V-a

Problema 1 – Case

Păcală, tocmai a fost ales primar în satul său, motiv pentru care şi-a luat rolul în serios şi dorind să aibă o evidenţă clară a sătenilor a numerotat casele din sat astfel încât să ştie câţi bărbaţi, femei şi copii locuiesc în fiecare casă.

Astfel, toate casele au un număr format din 3 cifre, unde prima cifră (de la stânga la dreapta) reprezintă numărul de bărbaţi ce locuiesc în acea casă (pot fi maxim 9 bărbaţi), a doua cifră reprezintă numărul de femei (pot fi maxim 9 femei), şi în fine ultima cifră a numărului reprezintă numărul de copii (maxim 9) ce aparţin familiei din acea casă.

Cerinţă

Cunoscându-se numărul de case din satul lui Păcală, precum şi numerele acestora să se determine:

- numărul total al sătenilor din sat.
- numărul minim de membrii ai unei familii.
- câte familii au număr minim de membrii.

Date de intrare

De la tastatură se citesc n, numărul de case şi apoi n numere naturale de 3 cifre, care reprezintă numerele caselor.

Date de ieşire

Pe primul rând de ecran se va afişa numărul total al sătenilor.

Pe al doilea rând al ecranului se va afişa numărul minim de membrii ai unei case.

Pe al treilea rând al ecranului se va afişa numărul familiilor cu număr minim de membri.

Restricţii şi precizări

- numărul de case este mai mic decât 100

- în fiecare casă există cel puţin un bărbat şi cel puţin o femeie.

Exemplu

Date de intrare	Explicaţie	Date de ieşire
6 232 215 340 325 450 124	6 case casa 1 : 2 bărbaţi, 3 femei, 2 copii casa 2 : 2 bărbaţi, 1 femei, 5 copii casa 3 : 3 bărbaţi, 4 femei, 0 copii casa 4 : 3 bărbaţi, 2 femei, 5 copii casa 5 : 4 bărbaţi, 5 femei, 0 copii casa 6 : 1 bărbaţi, 2 femei, 4 copii Numărul total al sătenilor : 48 Numărul minim de membrii : 7 Numărul de familii cu număr minim : 3 (familiile din casele 1, 3 şi 6)	48 7 3

Timp maxim de execuţie/test: 1 secundă

Rezolvare:

```
program jcase;
var n, nb,nf,nc,i,nrc,c1, c2, c3, nrl, min,
npc,nrcmin:integer;
begin
 nb:=0;
 min:=32000 ;
 npc:=0;
 nf:=0;
 nc:=0;
 nrl:=0;
 nrcmin:=0;
 write('numarul de case din sat='); readln(n);
 for i:= 1 to n do
 begin
  write ('nr casei=');readln(nrc);
  c1:=(nrc) mod 10;
  nc:=nc+c1;
  nrc:= nrc div 10;
  c2:=(nrc) mod 10;
  nf:=nf+c2 ;
```

```
  nrc:= nrc div 10;
  c3:=(nrc) mod 10 ;
  nb:=nb+c3 ;
  nrl:=nb+nf+nc ;
  npc:=c1+c2+c3;
  if npc<min then begin min:=npc; nrcmin:=1; end
      else if npc=min then nrcmin:=nrcmin+1;
 end;
 writeln('Numarul de barbari este egal cu ',nb,
'.');
 writeln('Numarul de femei este egal cu ',nf,
'.');
 writeln('Numarul de copii este egal cu ',nc,
'.');
 writeln('Numarul de locuitori ai satului este '
,nrl, '.');
 writeln('Numarul minim de locuitori ai unei
case este ' ,min, '.');
 writeln('Numarul de case ce au numar minim de
          locuitori este ', nrcmin,'.');
end.
```

Problema 2 – Vrăji

La Şcoala de Vrăjitorie Hogswarts, Harry Potter şi colegii săi îşi pun la încercare puterea vrăjilor cu ajutorul baghetelor magice. O vrajă constă în mutarea unuia sau a mai multor obiecte din încăperile şcolii în 'camera vrăjilor' unde se află adunaţi toţi elevii.

Fiecare dintre cei n 'elevi vrăjitori' este înzestrat cu o anumită putere; dacă un elev are puterea 1, cu o vrajă el aduce 1 obiect, dacă puterea este 2 cu o vrajă el va aduce 2 obiecte, … pentru un elev cu puterea de valoare p, cu o vrajă el va aduce p obiecte. Pe de altă parte, fiecare elev are o anumită rapiditate (viteză) de efectuare a vrăjilor. Astfel, pe parcursul unei ore, un elev cu viteza 1 va reuşi să facă o singură vrajă, un elev cu viteza 2 va reuşi două vrăji una după alta etc. Evident, un elev cu puterea 3 şi care are

viteza 4, va reuşi să aducă până la sfârşitul orei 12 obiecte (3 la prima vrajă, încă 3 la a doua vrajă, încă 3 la a treia vrajă şi încă 3 la ultima vrajă).
La sfârşitul orei de vrăjitorie, fiecare elev primeşte un număr de cutii pentru a ambala în ele numai obiectele aduse de el, astfel încât în fiecare dintre cutiile sale să se afle acelaşi număr de obiecte. Profesorul Dumbledore vrea în plus ca fiecare elev să primească acelaşi număr de cutii. O soluţie simplă ar fi să distribuie fiecărui elev o singură cutie, însă el şi-ar dori să distribuie cât mai multe cutii.

Cerinţă:
Cunoscând pentru fiecare dintre cei n 'elevi vrăjitori' ai şcolii Hogswarts, puterea cu care este înzestrat şi viteza cu care reuşeşte să facă vrăjile, determinaţi:
- cel mai mare număr de obiecte ce pot fi aduse până la sfârşitul orei de către un singur 'elev vrăjitor'
- care este numărul maxim de cutii pe care le va primi fiecare elev ţinând cont de faptul că fiecare elev va trebui să îşi distribuie în mod egal obiectele sale în aceste cutii;

Date de intrare:
De la tastatură se citesc:
- n numărul elevilor,
- pentru fiecare elev se vor citi de pe un rând de ecran 2 numere despărţite prin spaţiu reprezentând puterea sa şi viteza sa (în această ordine)

Date de ieşire:
Se vor afişa pe ecran:
- pe primul rând, cel mai mare număr de obiecte ce pot fi aduse în 'camera vrăjilor' de către un singur 'elev vrăjitor' la sfârşitul orei
- pe al doilea rând, cel mai mare număr de cutii pe care îl poate primi fiecare elev respectând condiţiile din problemă

Restricţii şi precizări:

- Numărul n al elevilor, puterea şi viteza fiecăruia sunt numere naturale mai mari decât zero şi mai mici sau egale cu 100
- Fiecare cutie va conţine numai obiecte ale unui singur 'elev vrăjitor'
- Fiecare elev va primi acelaşi număr de cutii

Verificaţi să NU aveţi în programul vostru uses crt sau #include <conio.h>

Exemple:

Date de intrare	**Date de ieşire**	**Explicaţie**
5 **5 2** **6 4** **3 10** **20 2** **7 2**	**40** **2**	5 - 'elevi vrăjitori' elev 1 : Număr total de obiecte =5*2=10 elev 2 : Număr total de obiecte=6*4=24 elev 3 : Număr total de obiecte=3*10=30 elev 4 : Număr total de obiecte=20*2=40 elev 5 : Număr total de obiecte=7*2=14 **40** este cel mai mare număr de obiecte aduse de un vrăjitor **2** este cel mai mare număr de cutii pe care îl poate primi fiecare elev
Date de intrare	**Date de ieşire**	**Explicaţie**
3 **4 2** **6 8** **6 6**	**48** **4**	3 - 'elevi vrăjitori' elev 1 : Număr total de obiecte =4*2=8 elev 2 : Număr total de obiecte=6*8=48 elev 3 : Număr total de obiecte=6*6=36 **48** este cel mai mare număr de obiecte aduse de un vrăjitor **4** este cel mai mare număr de cutii pe care îl poate primi fiecare elev

Timp de rulare/test: 1 secundă

Rezolvare:

```
program vraji;
var ne,i, p,v,nro,nrcut,
    max,j,min:integer;
    f,g:text;
    nu_se_imparte:boolean;
    vo:array[1..100] of integer;
begin
 assign(f,'vraji.in');
 reset(f);
 readln(f,ne);
 max:=0;
 for i:=1 to ne do
 begin
  readln(f,p,v);
  nro:=p*v;
  if nro>max then max:=nro;
  vo[i]:=nro;
  min:=3200;
  if min>nro then min:=nro;
 end;
 for i:=1 to min do
   begin
    nu_se_imparte:=false;
    for j:=1 to ne do
         if vo[j] mod i<>0 then
                    nu_se_imparte:=true;
    if nu_se_imparte=false then nrcut:=i;
   end;
 close(f);
 assign(g,'vraji.out');rewrite(g); writeln
(g,max);
 write(g,nrcut);
 close(g);
 writeln('max=',max);
end.
```

1.5. JOCURI CELEBRE CU NUMERE: NIM ŞI SUDOKU

Toată lumea, de la copii, până la adulţi, cunoaşte celebrele jocuri X şi 0, spânzurătoarea, războiul navelor, biscuitele etc. Sau poate aşa cred eu? Chiar dacă pentru majoritatea copiilor din ziua de azi pare greu de crezut, cei mai mulţi dintre noi (generaţiile de până în 1985) am crescut petrecând mult timp cu aceste jocuri, scriind şi desenând pe multe hârtii.

Chiar dacă astăzi copiii petrec foarte mult timp în faţa calculatorului, oferind o atenţie mai mare jocurilor în detrimentul altor activităţi precum cititul, jocurilor în aer liber, ele, jocurile pe calculator reuşesc să dezvolte destule abilităţi ca atenţia distributivă, îndemânarea, rapiditatea în mişcări, spiritul de observaţie, logica şi multe altele.

Faţă de alte jocuri pe calculator, care influenţează negativ caracterul copilului, jocurile de logică, au scopul de a-i face pe cei mici să gândească şi de a duce la bun sfârşit o acţiune începută. Nu mai este nevoie să descărcăm jocuri de pe Internet, deoarece se pot juca online, singur, partener fiind calculatorul, sau împreună cu prietenii, reprezentând un mod de relaxare şi o cale de a ne întoarce în timp, de dragul vremurilor trecute.

Am văzut că sunt multe beneficii ale jocurilor de logică dacă le folosim, se dezvoltă însă mai multe abilităţi, avem şi mari satisfacţii, dacă le creăm noi.

Cu elevii capabili de performanţă şi dornici să înveţe programare se poate încerca crearea unor jocuri ca X şi O, Nim şi Sudoku (joc care apare des în revistele copiilor) în limbajul C++/Pascal.

Jocul NIM

Jocul Nim este unul dintre cele mai cunoscute jocuri imparţiale şi este originar din China. În acest joc se consideră un anumit număr de grămezi şi fiecare grămadă având un anumit număr de piese. La fiecare pas jucătorul aflat la mutare trebuie să extragă un număr nenul de piese dintr-o singură grămadă (se poate să se extragă şi toate piesele dintr-o grămadă). La acest joc câştigă cine extrage ultimele piese aflate în joc.

De obicei jocul Nim se joacă cu 3 grămezi, dar indiferent de numărul de grămezi, strategia rămâne aceeaşi.

Nim este un joc simplu cu posibilităţi finite, spre deosebire de şah. De fapt Nim nu este cu mult mai complex decât un puzzle. Spre deosebire totuşi de celelalte jocuri cu posibilităţi finite (de exemplu "*X şi* 0") există o mare varietate de jocuri Nim, obiectele putând fi de orice fel: monede, pietre, portocale, etc. De asemenea

se pot pune oricâte obiecte într-o grămadă şi pot fi oricâte grămezi. Ca şi "X şi 0", Nim este o provocare până în momentul în care jucătorul îşi dă seama de strategia câştigătoare. În cazul acestui joc nu există remiză şi mai mult, dacă primul jucător ştie strategia câştigătoare şi aplică corect algoritmul nu poate fi învins.

Pentru a determina strategia de câştig ne vom folosi de operaţia xor (exclusive or) şi proprietăţile ei. Această operaţie se realizează prin operatorul ^ în limbajul C/C++, şi prin xor în Pascal. Ca operaţie pe biţi, ea poate fi interpretată ca adunare în baza 2 fără transport, dupa cum reiese din tabelul următor.

Operaţia xor (sau exclusiv)

p1	p2	p1 xor p2
0	0	0
0	1	1
1	0	1
1	1	0

Atunci când facem xor între numere naturale trebuie, mai întâi, să transformăm aceste numere în baza 2, şi apoi să adunăm, fără transport, biţii de pe fiecare poziţie. De exemplu, 1 xor 7 xor 5 = 3, deoarece avem:

$1 = (001)_2$ xor

$7 = (111)_2$

$\underline{5 = (101)_2}$

$(011)_2 = 3$

Strategia de câştig

Pe baza operaţiei xor, strategia de câştig pentru NIM poate fi formulată conform următoarei teoreme:

Fie N grămezi. Prima grămadă are x_1 pietre, cea de a doua x_2, şi aşa mai departe, până la ultima care are x_n pietre. O astfel de poziţie este pierzătoare în jocul de NIM dacă şi numai dacă suma-xor a numerelor de pietre din grămezi este 0, adica daca x_1 xor x_2 ... xor $x_n = 0$.

Putem demonstra această teorema arătând că dintr-o stare cu suma-xor egală cu 0 (pierzătoare), oricum am muta, nu putem ajunge decât într-o stare cu suma-xor nenulă (câştigătoare), şi că dintr-o stare câştigătoare putem efectua o mutare în mod convenabil astfel încât să ajungem într-o stare de pierdere.

Exemplu:

Am ales un exemplu pentru varianta clasică, cu 3 grămezi. În fiecare grămadă avem: 3, 5 şi respectiv 7 monede. Voi construi o matrice, care pe linia i (i intre 1 si 3) va avea numărul de obiecte din grămadă *i* scris în binar. În acest caz matricea va fi următoarea:

Grămada 1 (3 monede)	0 1 1
Grămada 2 (5 monede)	1 0 1
Grămada 3 (7 monede)	1 1 1
	0 0 1

După ce aplicăm XOR pe fiecare coloană se poate observa că pe coloana 3 se obţine 1. Ideea algoritmului este să luăm atâtea obiecte dintr-o anumită grămadă astfel încât efectuând din nou XOR pe fiecare coloană să obţinem 0. În cazul nostru dacă înlocuim prima linie cu 0 1 0 (de fapt nu este nevoie să înlocuim toată linia ci doar elementele de pe coloanele care dau 1) şi facem XOR pe toate coloanele vom obţine 0, deci vom ajunge într-o **poziţie sigură**. Atunci când modificăm linia din matrice se modifică şi numărul de elemente din matrice... în cazul nostru pe linia 1 ar trebui să avem 2 elemente, deci trebuie să luăm o monedă din prima grămadă.

Matricea binara arată astfel:

Grămada 1 (2 monede)	0	1	0
Grămada 2 (5 monede)	1	0	1
Grămada 3 (7 monede)	1	1	1
	0	0	0

Voi prezenta mai jos toate etapele acestui exemplu:

În grămada 1 au rămas 2 obiecte şi calculatorul ia un obiect din grămada1. După mutarea calculatorului, matricea binară va arăta astfel:

Grămada 1 (1 monede)	0	0	1
Grămada 2 (5 monede)	1	0	1
Grămada 3 (7 monede)	1	1	1
	0	1	1

Efectuând XOR-ul se va constata că pe coloanele 2 şi 3 vom avea 1, deci trebuie să luăm monede astfel încât să obţinem

0. Înlocuim ultima linie cu 1 0 0 (luăm 3 monede astfel încât în ultima grămadă să rămână 4 obiecte).

Grămada 1 (1 monede)	0 0 1
Grămada 2 (5 monede)	1 0 1
Grămada 3 (4 monede)	1 0 0
	0 0 0

În grămada 3 au rămas 4 obiecte şi calculatorul ia un obiect din grămada2. Matricea binara arată astfel:

Grămada 1 (1 monede)	0 0 1
Grămada 2 (4 monede)	1 0 0
Grămada 3 (4 monede)	1 0 0
	0 0 1

Efectuând XOR-ul se va constata că pe coloana 3 vom avea 1. Înlocuim linia1 cu 0 0 0 (luăm ultima monedă din grămada 1).

Grămada 1 (0 monede)	0 0 0
Grămada 2 (4 monede)	1 0 0
Grămada 3 (4 monede)	1 0 0
	0 0 0

În prima grămadă nu a rămas nici un obiect şi calculatorul ia un obiect din grămada 2. Matricea binară arată astfel:

Grămada 1 (0 monede)	0 0 0
Grămada 2 (3 monede)	0 1 1
Grămada 3 (4 monede)	1 0 0
	1 1 1

Efectuând XOR-ul se va constata ca pe toate coloanele vom avea 1. Înlocuim linia 3 cu 0 1 1 (luăm monede din grămada 3 astfel încât să rămână 3 obiecte).

Grămada 1 (0 monede) 0 0 0
Grămada 2 (3 monede) 0 1 1
Grămada 3 (3 monede) 0 1 1
0 0 0

În grămada 3 au rămas 3 obiecte şi calculatorul ia un obiect din grămada 2. Matricea binara arată astfel:

Grămada 1 (0 monede) 0 0 0
Grămada 2 (2 monede) 0 1 0
Grămada 3 (3 monede) 0 1 1
0 0 1

Efectuând XOR-ul se va constata că pe coloana 3 vom avea 1. Înlocuim linia3 cu 0 1 0 (luăm o monedă din grămada 3).

Grămada 1 (0 monede) 0 0 0
Grămada 2 (2 monede) 0 1 0
Grămada 3 (2 monede) 0 1 0
0 0 0

În grămada 3 au rămas 2 obiecte şi calculatorul ia un obiect din grămada 2. Matricea binara arată astfel:

Grămada 1 (0 monede) 0 0 0
Grămada 2 (1 monede) 0 0 1
Grămada 3 (2 monede) 0 1 0
0 1 1

Efectuând XOR-ul se va constata că pe coloanele 2 şi 3 vom avea 1. Înlocuim linia3 cu 0 0 1 (luăm o monedă din grămada 3).

Grămada 1 (0 monede) 0 0 0
Grămada 2 (1 monede) 0 0 1
Grămada 3 (1 monede) 0 0 1
0 0 0

În grămada 3 a rămas un obiect şi calculatorul ia un obiect din grămada 2. Matricea binara arată astfel:

Grămada 1 (0 monede)	0 0 0
Grămada 2 (0 monede)	0 0 0
Grămada 3 (1 monede)	0 0 1
	0 0 1

A rămas o singură monedă şi e rândul meu... Am câştigat!!

Iată programul pt. Jocul NIM:

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <stdlib.h>
const n=3;          //nr. gramezi
int G[n]={3,5,7};  //configuratie initiala
gramezi
int J[n];           //configuratia curenta

void afisare()
{
  int i;
  cout<<"\nN I M";
  cout<<"\n--------\n";
  for(i=0;i<n;i++)
    cout<<"\n
Gramada"<<setw(2)<<i+1<<")."<<J[i]<<"\n";
}

void mutajuc()   //mutare jucator
{
   int g,p;
   do
   {   cout<<"\n >> Mutare [G,p] ? ";
```

```
    //G=gramada din care se iau piese; p=nr.
piese luate
       cin>>g>>p;
   }
   while ( g<=0 || g>n || p<=0 || p>J[g-1]);
   J[g-1]-=p;        //MUTARE: jucatorul ia p
piese din gramada g
}

int sumaXOR(int v[], int n)
//suma XOR a numerelor v[0],...,v[n-1]
{
  int X=v[0];
  for(int i=1;i<n;i++)X^=v[i];
  return X;
}

void mutacalc(int v[], int n)
//mutarea calculatorului
{
  int g,p;
  int x=sumaXOR(v,n);
  if(x)
      //suma XOR este diferita de 0, deci
calculatorul castiga
//luand atatea piese dintr-o gramada g a.i. suma
XOR sa devina 0
    for(g=0;g<n;g++)
    {
      int r=v[g]^x;
      if(v[g]>r)
      { p=v[g]-r;
        v[g]-=p;     //MUTARE: jucatorul ia p
piese din gramada i
        break;
      }
    }
   else
```

```
      //suma XOR este 0, calculatorul joaca la
intamplare
      //luand un nr. aleator de piese dintr-o
gramada g
   {  do
      {  g=random(n);          }
      while (v[g]==0);
      p=random(v[g])+1;
      v[g]-=p;       //MUTARE: jucatorul ia p
piese din gramada g
   }
}

int sumar()       //suma pieselor ramase
{
  int s=0;
  for(int i=0;i<n;i++) s+=J[i];
  return s;
}
void main()
{
  int i=0;
  clrscr();                    //  Incepe JOCUL
NIM
  for(i=0;i<n;i++) J[i]=G[i];  // configuratia
initiala
  afisare();
  while (1)
  {
      mutajuc();
      afisare();
      if(sumar()==0)
      {   cout<<"\n  FELICITARI! AI CASTIGAT
ACEST JOC!\n";
          getch();
          break;
      }
      cout<<"\n Mutare calculator...\n";
```

```
        mutacalc(J,n);
        afisare();
        if(sumar()==0)
        {     cout<<"\n  AI PIERDUT ACEST JOC !\n";
              getch();
              break;
        }
   }
}
```

Jocul SUDOKU

Este un joc popular al zilelor noastre, majoritatea ziarelor propun o grilă Sudoku în pagina lor de jocuri.

De obicei, jocul este propus sub forma unei grile de 9×9, împărţită în sub-grile de 3×3, numite "regiuni". Câteva celule conţin cifre, aşa-numitele "căsuţe precompletate". Scopul este acela de a umple celulele goale, cu câte o cifră în fiecare celulă, în aşa fel încât fiecare rând, fiecare coloană şi fiecare regiune să conţină cifrele de la 1 la 9 o singură dată. În consecinţă, fiecare cifră a soluţiei apare o singură dată în trei "direcţii", de unde numele "cifră unică". Când o cifră poate fi înscrisă într-o celulă, aceasta se numeşte candidată.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

O grilă 9×9 de Sudoku

Specialiştii recomandă practicarea jocului Sudoku ca antrenament pentru gândirea logică. Nivelul de dificultate poate în acest caz să fie adaptat publicului.

Pentru un informatician, a programa căutarea unei soluţii prin intermediul contingenţelor sau multiplelor contingenţe (aşa cum se cere pentru problemele cele mai dificile) este o sarcină relativ simplă. Un astfel program imită un jucător uman care caută o soluţie fără să caute la întâmplare.

Este de asemenea relativ simplu conceperea unui algoritm de căutare prin backtracking. De obicei, este suficient ca algoritmul să aleagă 1 pentru prima celulă, apoi 2 pentru următoarea, atâta timp cât nu apare nici o contradicţie. Când apare o astfel de contradicţie, algoritmul încearcă o altă valoare pentru căsuţa care duce la contradicţie. O dată epuizate toate posibilităţile pentru această căsuţă, algoritmul "revine" şi reîncepe cu penultima celulă.

Chiar dacă acest algoritm nu este foarte eficace, el va găsi o soluţie dacă dispune de suficient timp. O grilă de 9×9 este de obicei rezolvată în mai puţin de trei secunde de un computer modern care are acces la un interpretor, şi în mai puţin timp cu un limbaj compilat.

Un program mai eficace se va baza pe cifre candidate potenţiale pentru fiecare căsuţă, eliminând cifrele candidat imposibile până când rămâne o singură cifră. Cunoscând această cifră, algoritmul poate găsi o altă cifră pentru o altă căsuţă, şi tot aşa.

O alternativă la backtracking este aceea de a recurge la metodele preconizate de programarea logică, aşa cum este implementată de Prolog şi de Scheme. În acest caz, se furnizează programului constrângerile grilei (o cifră pe rând, pe coloană şi pe regiune; cifrele descoperite); acest program va lua singur deciziile pentru rezolvarea problemei. Ştiind că majoritatea grilelor au o soluţie unică, căutarea va avea cu siguranţă succes.

Donald Knuth a pus la punct un algoritm care face apel la listele dublu înlănţuite, şi care se pare că este foarte eficace pentru rezolvarea acestui tip de problemă. S-a demonstrat că acest algoritm este indicat pentru rezolvarea unui Sudoku, durând doar câteva milisecunde. Datorită vitezei sale, este acum preferat de majoritatea programatorilor.

Propun rezolvarea Sudok-ului prin două metode, prima prin tehnica Greedy şi a doua prin metoda Backtracking.

Varianta 1, specifică metodei de programare Greedy.

```
// Rezolvarea jocului SUDOKU, prin aplicarea
repetata a uneia din // urmatoarele doua reguli:
//  1. daca pt. un element S[x][y] exista un
singur nr. valid k, // atunci luam S[x][y]=k
//  2. daca pt. un bloc (linie, coloana sau
patrat 3x3) un nr. k // este valid pt. un singur
element S[x][y],atunci luam S[x][y]=k
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
int A[9][9]; //configuratia initiala a jocului
int S[9][9]; //solutia jocului
int V[9][9][9];
//V[x][y][k-1]=1 daca si numai daca k este valid
pt. S[x][y]
int nr; //nr. de elemente completate

int valid(int x,int y,int k)
//verificam daca k este valid pt. S[x][y]
{ int i,j;
  // verificam existenta lui k pe linia x
  for (i=0;i<9;i++)
   if (S[x][i]==k) return 0;
  // verificam existenta lui k pe coloana y
  for (i=0;i<9;i++)
   if (S[i][y]==k) return 0;
  // verificam existenta lui k in patratul 3x3
in care se afla
  // S[x][y]
   int l,c;
   l=3*(x/3); c=3*(y/3);
```

```
   for (i=l;i<l+3;i++)
    for (j=c;j<c+3;j++)
     if (S[i][j]==k) return  0;
   return 1;
}

void afisare(int A[9][9])
{ int i,j;
  cout<<"|---------|---------|---------|"<<endl;
  for (i=0;i<9;i++)
  { cout<<"|";
    for (j=0;j<9;j++)
    {  if (A[i][j]) cout<<" "<<A[i][j]<<" ";
       else cout<<"   ";
       if (j%3==2) cout<<"|";
    }

cout<<endl;
    if (i%3==2) cout<<"|---------|---------|----
-----|"<<endl;
  }
}

void main()
{
 int i,j,k,nk,k0,l,c,i1,j1,i0,j0,ap;
 cout<<"Configuratia initiala:"<<endl;
 ifstream f("sudoku1.txt");
 for (i=0;i<9;i++)
  for (j=0;j<9;j++) f>>A[i][j];
 f.close ();
 afisare(A);getch();
 for (i=0;i<9;i++)
  for (j=0;j<9;j++) S[i][j]=A[i][j];
   //calculam tabloul V
 for (i=0;i<9;i++)
  for (j=0;j<9;j++)
   for (k=1;k<10;k++)
```

```
    if (S[i][j]==0) V[i][j][k-1]=valid(i,j,k);
    else V[i][j][k-1]=0;
 nr=0;//calculam nr. de elemente completate
 for (i=0;i<9;i++)
  for (j=0;j<9;j++)
   if (S[i][j]) nr++;
   //aplicam repetat, cat timp este posibil, o
regula de tipul 1
   //sau 2
 int ex=1;
//ex devine 0 cand nu mai putem aplica o astfel
de regula
 int r=0;//r devine 1 in una din urmatoarele
doua situatii:
//   1. pt. un element S[i][j]  nu exista niciun
nr. valid
//   2. pt. un bloc (linie, coloana sau patrat
3x3) un nr. k
//(ce nu este element in bloc) nu este valid pt.
niciun element
// deci in ambele cazuri problema nu are solutie
 while (ex)
 { ex=0;
   for (i=0;i<9;i++)  //incercam aplicarea unei
reguli de tipul 1
     for (j=0;j<9;j++)
      if (S[i][j]==0)
      { nk=0;//nr. de numere k valide pt.
S[i][j]
        for (k=1;k<10;k++)
       if (V[i][j][k-1])
       { nk++;
         k0=k;//retinem numarul valid pt. S[i][j]
       }
        if (nk==1) //aplicam regula de tipul 1
      { S[i][j]=k0; //actualizam variabilele
utilizate
       afisare(S);getch();
```

```
        nr++;
        ex=1;
        for (c=0;c<9;c++)  V[i][c][k0-1]=0;
        for (l=0;l<9;l++)  V[l][j][k0-1]=0;
           l=3*(i/3); c=3*(j/3);
           for (i1=l;i1<l+3;i1++)
             for (j1=c;j1<c+3;j1++) V[i1][j1][k0-
1]=0;
      }else
       if (nk==0) r=1;
       }
   if (ex==0)
   {  for (k=1;k<10;k++)
//incercam aplicarea unei reguli de tipul 2 pt.
k
      {
        for (i=0;i<9;i++) //analizam linia i
      { ap=0;
//ap devine 1 daca nr. k este element al acestei
linii
          for (j=0;j<9;j++)
         if (S[i][j]==k) ap=1;
        if (ap==0)
        { nk=0;
//nr. de elemente de pe linia i pt. care k este
valid
         for (j=0;j<9;j++)
          if ((S[i][j]==0) && (V[i][j][k-1]))
          { nk++;
            j0=j;
//retinem coloana elementului pt care k este
valid
          }
            if (nk==1) //aplicam regula de tipul
2
         { S[i][j0]=k; //actualizam variabilele
utilizate
           afisare(S);getch();
```

```
            nr++;
            ex=1;
            for (c=0;c<9;c++)  V[i][c][k-1]=0;
            for (l=0;l<9;l++)  V[l][j0][k-1]=0;
               l=3*(i/3); c=3*(j0/3);
               for (i1=l;i1<l+3;i1++)
             for (j1=c;j1<c+3;j1++) V[i1][j1][k-
1]=0;
          }
          else
           if (nk==0) r=1;
        }
      }
       for (j=0;j<9;j++) //analizam coloana j
      { ap=0;
//ap devine 1 daca nr. k este element al acestei
coloane
          for (i=0;i<9;i++)
         if (S[i][j]==k) ap=1;
        if (ap==0)
        { nk=0;
//nr. de elemente de pe coloana j pt. care k
este valid
          for (i=0;i<9;i++)
           if ((S[i][j]==0) && (V[i][j][k-1]))
           { nk++;
             i0=i;
//retinem linia elementului pt care k este valid
           }
            if (nk==1) //aplicam regula de tipul
2
          { S[i0][j]=k; //actualizam variabilele
utilizate
            afisare(S);getch();
            nr++;
            ex=1;
            for (c=0;c<9;c++)  V[i0][c][k-1]=0;
            for (l=0;l<9;l++)  V[l][j][k-1]=0;
```

```
              l=3*(i0/3); c=3*(j/3);
              for (i1=l;i1<l+3;i1++)
             for (j1=c;j1<c+3;j1++) V[i1][j1][k-
1]=0;
          }
          else
           if (nk==0) r=1;
        }
      }
      for (l=0;l<9;l=l+3)
//analizam patratul 3x3 ce incepe pe linia l si
coloana c
      for (c=0;c<9;c=c+3)
      { ap=0;
//ap devine 1 daca nr. k este element al acestui
patrat
        for (i=l;i<l+3;i++)
        for (j=c;j<c+3;j++)
         if (S[i][j]==k) ap=1;
        if (ap==0)
        { nk=0;
//nr. de elemente din patratul 3x3 analizat pt.
care k este valid
          for (i=l;i<l+3;i++)
          for (j=c;j<c+3;j++)
           if ((S[i][j]==0) && (V[i][j][k-1]))
           { nk++;
             i0=i; j0=j;
//retinem linia si coloana elementului pt care k
este valid
           }
            if (nk==1) //aplicam regula de tipul
2
          { S[i0][j0]=k; //actualizam variabilele
utilizate
            afisare(S);getch();
            nr++;
            ex=1;
```

```
            for (j=0;j<9;j++)  V[i0][j][k-1]=0;
            for (i=0;i<9;i++)  V[i][j0][k-1]=0;
               for (i=l;i<l+3;i++)
             for (j=c;j<c+3;j++) V[i][j][k-1]=0;
          }
          else
           if (nk==0) r=1;
        }
      }
    }
   }
 }
 if (nr==81)
 { cout<<"Solutia finala:"<<endl;
   afisare(S);
 }
 else
  if (r) cout<<"Nu exista solutie!"<<endl;
  else
  { cout<<"Solutie partiala (de la care nu putem
continua doar prin aplicarea regulilor 1 si
2):"<<endl;
    afisare(S);
  }
}
```

Varianta 2, specifică metodei de programare Backtracking.

```
//Rezolvarea jocului SUDOKU, prin metoda
BACKTRACKING
#include <iostream.h>
#include <fstream.h>
int A[9][9]; //configuratia initiala a jocului
int S[9][9]; //solutia jocului

int valid(int x,int y,int v)
//verificam daca putem avea S[x][y]=v
{ int i,j;
  // verificam existenta lui v pe linia x
  for (i=0;i<9;i++)
   if (S[x][i]==v) return 0;
  // verificam existenta lui v pe coloana y
  for (i=0;i<9;i++)
   if (S[i][y]==v) return 0;
  // verificam existenta lui v in patratul 3x3
in care se afla
  // S[x][y]
   int l,c;
   l=3*(x/3); c=3*(y/3);
   for (i=l;i<l+3;i++)
    for (j=c;j<c+3;j++)
     if (S[i][j]==v) return  0;
   return 1;
}

int back(int x,int y)
//completam matricea S de la elementul S[x][y],
RECURSIV
{ int i,j,r;
```

```
  if (S[x][y]==0)  //elementul S[x][y] este
necompletat, deci
                   // incercam completarea lui
  { for (i=1;i<10;i++)
      if (valid(x,y,i))
//putem avea S[x][y]=i, deci efectuam aceasta
atribuire
//si continuam completarea matricei S, RECURSIV
      { S[x][y]=i;
      if (x==8 && y==8) return 1;
           else
            if (y==8)
       { if (back(x+1,0)) return 1;
            }
            else
             if (back(x,y+1)) return 1 ;
      }
    if (i==10)
//nu exista valoare valida pentru elementul
S[x][y],
//deci matricea curenta S nu conduce la solutie
    { if (A[x][y]==0) S[x][y]=0;
//refacem matricea curenta S,
//elementul  S[x][y] redevine necompletat
      return 0;
    }
  }
  else
//elementul S[x][y] este completat, deci
continuam completarea
// matricei S, RECURSIV
  { if (x==8 && y==8) return 1;
    else
```

```
     if (y==8)
     { if (back(x+1,0)) return 1;
     }
     else
       if (back(x,y+1)) return 1;
  }
}

void afisare(int A[9][9])
{ int i,j;
  cout<<"|---------|---------|---------|"<<endl;
  for (i=0;i<9;i++)
  { cout<<"|";
    for (j=0;j<9;j++)
    {  if (A[i][j]) cout<<" "<<A[i][j]<<" ";
       else cout<<"   ";
       if (j%3==2) cout<<"|";
    }
    cout<<endl;
    if (i%3==2) cout<<"|---------|---------|----
-----|"<<endl;
  }
}

void main()
{
 int i,j;
 cout<<"Configuratia initiala:"<<endl;
 ifstream f("sudoku1.txt");
 for (i=0;i<9;i++)
  for (j=0;j<9;j++) f>>A[i][j];
 f.close ();
 afisare(A);
```

```
 for (i=0;i<9;i++)
  for (j=0;j<9;j++) S[i][j]=A[i][j];
 if (back(0,0))
 { cout<<"Solutia finala:"<<endl;
   afisare(S);
 }
 else cout<<"Nu exista solutie!"<<endl;
}
```

Prima variantă este specifică metodei Greedy de programare. Ea este eficientă din punct de vedere al timpului de execuţie şi rezolvă jocul doar prin aplicarea repetată a regulilor 1 si 2. De exemplu, pentru următoarea configuraţie iniţială a jocului

```
-----------------------
|3 0 6|9 0 0|0 0 2|
|2 0 0|3 7 8|0 0 9|
|7 0 0|5 0 6|1 0 3|
-----------------------
|6 2 0|0 0 3|9 1 0|
|1 0 0|7 0 5|0 0 0|
|4 7 8|1 0 2|0 6 0|
-----------------------
|0 0 0|0 5 0|0 3 0|
|8 6 0|0 1 0|0 5 0|
|0 0 2|0 0 0|8 0 0|
-----------------------
```

rezolvarea acestuia, folosind această variantă, este:

```
-----------------------
|3 8 6|9 4 1|5 7 2|
|2 5 1|3 7 8|6 4 9|
|7 9 4|5 2 6|1 8 3|
```

```
-------------------------
| 6 2 5 | 4 8 3 | 9 1 7 |
| 1 3 9 | 7 6 5 | 4 2 8 |
| 4 7 8 | 1 9 2 | 3 6 5 |
-------------------------
| 9 1 7 | 8 5 4 | 2 3 6 |
| 8 6 3 | 2 1 9 | 7 5 4 |
| 5 4 2 | 6 3 7 | 8 9 1 |
-------------------------
```

În schimb, pentru următoarea configuraţie iniţială a jocului

```
-------------------------
| 1 0 0 | 0 0 4 | 0 0 0 |
| 0 0 5 | 3 0 0 | 2 0 0 |
| 0 6 0 | 0 8 0 | 0 0 9 |
-------------------------
| 0 4 0 | 0 0 0 | 0 0 7 |
| 0 0 7 | 0 0 0 | 3 0 0 |
| 3 0 0 | 0 0 0 | 0 6 0 |
-------------------------
| 6 0 0 | 0 0 7 | 0 2 0 |
| 0 3 0 | 0 9 0 | 0 0 8 |
| 0 0 2 | 1 0 0 | 5 0 0 |
-------------------------
```

aplicarea variantei Greedy nu conduce la rezolvarea jocului, ci produce doar următoarea soluţie parţială (de la care nu mai putem continua doar prin aplicarea regulilor 1 si 2):

```
-------------------------
| 1 0 0 | 0 0 4 | 0 0 0 |
| 0 0 5 | 3 0 0 | 2 0 0 |
| 0 6 0 | 0 8 0 | 0 0 9 |
-------------------------
```

0 4 6	0 0 0	0 0 7
0 0 7	0 0 0	3 0 0
3 0 0	0 0 0	0 6 0
---
6 0 0	0 0 7	0 2 0
0 3 0	0 9 0	0 0 8
0 0 2	1 0 0	5 0 0

A doua variantă este specifică metodei Backtracking de programare. Ea este mai puţin eficientă din punct de vedere al timpului de execuţie, dar rezolvă întotdeauna jocul. Pentru prima configuraţie iniţială de mai sus, aplicarea variantei Backtracking conduce, evident, la aceeaşi soluţie finală ca şi în cazul aplicării metodei Greedy.

În schimb, pentru a doua configuraţie iniţială de mai sus, aplicarea variantei Backtracking conduce la rezolvarea acestuia:

---
1 9 8	7 2 4	6 3 5
4 7 5	3 6 9	2 8 1
2 6 3	5 8 1	7 4 9
---
9 4 6	2 3 5	8 1 7
8 2 7	9 1 6	3 5 4
3 5 1	4 7 8	9 6 2
---
6 1 9	8 5 7	4 2 3
5 3 4	6 9 2	1 7 8
7 8 2	1 4 3	5 9 6

CAPITOLUL 2
CREATIVITATEA ŞI GÂNDIREA CRITICĂ

2.1. CREATIVITATEA – PRECIZĂRI CONCEPTUALE

Omul a încercat întotdeauna să înţeleagă tainele lumii din afara şi dinăuntrul ei, iar atunci când nu a avut posibilitatea să le înţeleagă şi să le explice în mod ştiinţific a născocit o explicaţie potrivită concepţiei sale despre lume.

Existenţa unor diferenţe între oameni în ceea ce priveşte rezultatele calitative şi cantitative ale muncii lor, a fost remarcată din zorile umanităţii. Dezvoltarea şi ascensiunea societăţii nu a fost şi nu este posibilă fără contribuţia creatoare a oamenilor. Din acest motiv este firesc să considerăm activitatea creatoare drept cea mai înaltă formă a activităţii omeneşti.

CREATIVITATEA

Termenul de creativitate îşi are originea îi cuvântul latin „*creare*” care înseamnă „a zămisli”, „a făuri”, „a crea”, „a naşte”. Însăşi originea cuvântului demonstrează că termenul de creativitate defineşte un proces, un act dinamic care se dezvoltă şi se desăvârşeşte continuu şi cuprinde atât originea cât şi scopul.

Deşi creativitatea este veche de când lumea, abia în a doua jumătate a secolului nostru termenul de creativitate a fost larg asimilat în limbile de circulaţie internaţională.

La început, termenii de creativitate şi creativ erau folosiţi sub denumirea de: dotaţie, aptitudine, talent, imaginaţie creatoare şi chiar inteligenţă. Nici astăzi creativitatea nu s-a debarasat de sinonime ca: inteligenţă fluidă (R.B. Cattel), gândire divergentă (J.P. Guilford), gândire direcţionată creatoare (J. Bruner), imaginaţie creatoare (J. Piaget), gândire aventuroasă (F. Bartlett).

Conceptul de creativitate încă nu este clar definit deşi de el se ocupă literaţii, psihologii, pedagogii, sociologii sau chiar recent „*cercetătorii specialişti în creativitate”*.

I.A. Taylor consemnează că a întâlnit mai mult de o sută de definiţii ale noţiunii de creativitate.

Unii cercetători definesc creativitatea ca fiind: „facultatea de a introduce în lume un lucru oarecare nou” (Moreno).

În Dicţionarul de Psihologie „Larousse” această facultate este definită ca o dispoziţie de a crea ce există în stare potenţială la toţi indivizii de toate vârstele.

În definirea creativităţii trebuie să se aibă în vedere: realizarea unui produs original nou, de valoare şi util. Împărtăşind această opinie, Al. Roşca este de părere că „mai frecvent, creativitatea este considerată ca fiind un proces care duce la un

anumit produs, caracterizat prin originalitate sau noutate şi prin valoare sau utilitate pentru societate."[1]

Creativitatea este o capacitate (proprietate, dimensiune) complexă şi fundamentală a personalităţii, care, sprijinindu-se pe date sau produse anterioare, în îmbinarea cu investigaţii şi date noi, produce ceva nou, original, de valoare şi eficienţă ştiinţifică şi social-utilă, ca rezultat al influenţelor şi relaţiilor factorilor subiectivi şi obiectivi, adică a posibilităţilor şi calităţilor persoanei şi a condiţiilor ambientale ale mediului socio-cultural.

După alţi psihologi, privită într-un cadru mai larg, creativitatea se referă la găsirea de soluţii, idei, probleme, metode, care nu sunt noi pentru societate, dar la care s-a ajuns pe o cale independentă.

În psihologie, conceptul de creativitate are următoarele trei accepţiuni:

„1. de comportament şi activitate psihică creativă; 2. de structură a personalităţii şi stil creativ; 3. creativitate de grup, în care interacţiunile şi comunicarea mijlocesc generarea de noi idei, deci, duc la efecte creative."[2]

Ca proces de ordin psihologic "procesul creaţiei poate fi

[1] Al., Roşca, *Creativitatea,* Bucureşti, Editura Enciclopedică Română, 1972, p. 7;8.
[2] Paul, Popescu-Neveanu şi colaboratorii, *Psihologie*, Bucureşti, Editura Didactică şi Pedagogică, 1993, p. 178.

conceput ca o activitate mentală complexă de interferare şi intersectare operaţională a celor mai importante procese cognitive şi necognitive, cu finalitate rezolutivă şi cu urmări predicative pentru procesul social."[1]

În urma cercetărilor efectuate, I.C. Gowan şi G.D. Demas concluzionează: „Copiii sunt creativi în mod natural şi doar aşteaptă atmosfera propice pentru a-şi manifesta creativitatea."

Creativitatea este definită fie ca o capacitate, ca o caracteristică a performanţei persoanei, fie ca o capacitate de a descoperi sau a crea.

Activitatea creatoare poate fi considerată ca forma cea mai înaltă de manifestare a conduitei umane, care implică antrenarea unor multitudini de factori obiectivi şi subiectivi, în vederea producerii noului.

În cercetările sale, Taylor afirmă: „în primul rând psihologii sunt convinşi că toţi oamenii sunt într-o măsură creativi în mod potenţial". Indivizii se deosebesc în privinţa potenţialului creativ prin modul de exprimare a creativităţii lor.

Toţi oamenii sunt în diverse grade creativi şi numai unii din ei sunt talentaţi. Între conceptele de creativitate şi de talent nota comună este cea de originalitate. Dovedeşte talent cel ce

[1] N. C., Matei, *Probleme de psihologie şcolară*, Bucureşti, Editura Didactică şi Pedagogică, 1978, p. 154.

demonstrează o pregnantă originalitate. Deci, talentul corespunde creativităţii de nivel superior. Dar aceasta nu este exclusivă pentru că există şi o creativitate de nivel mediu şi una slabă, redusă. Conceptul nou de creativitate admite o mare contribuţie a influenţelor de mediu şi a educaţiei în formarea creativă a fiecăruia.

În mentalitatea modernă, creativitatea descărcată de mituri şi de tragismul implicat în ideea geniului neînţeles este concepută ca o activitate ce se desfăşoară într-un atelier de creaţie, ceea ce ne evocă materiale şi tehnici, instrumente şi priceperi, experienţă, capacitatea de a schimba prin aceasta „creaţia potenţială" în „creaţie virtuală".[1]

O ramură importantă a creativităţii o reprezintă creativitatea şcolară. Aceasta este diferită de creativitatea artistului; astfel, copilul de vârstă şcolară are o atitudine creatoare atunci când, pus în faţa unei sarcini şcolare, descoperă calea de rezolvare într-un mod personal.

Şcoala îl pune pe copil în faţa unor probleme noi, ajutându-l să înţeleagă diferitele relaţii dintre obiecte şi fenomene. Sensul creativităţii la elevi este acela de potenţial creativ de capacităţi aptitudinale predictive pentru performanţele creatoare

[1] Ursula, Şchiopu, *Creativitatea potenţială şi virtuală,* art.cit. Revista de psihologie nr.3, 1979.

de mai târziu.

Definirea şi explicarea fenomenului complex al creativităţii se sprijină şi pe încercările de a descifra factorii care o caracterizează, sensibilitatea faţă de problemele şi faţă de atitudinile şi sentimentele altora.

În şcoala generală nu putem vorbi de existenţa unei creativităţi absolute deoarece gândirea elevului se află în faza achiziţiilor fundamentale. Acum este perioada când dobândeşte şi îşi însuşeşte deprinderile elementare de muncă intelectuală, premisă a formării şi dezvoltării creativităţii.

Fiecare individ posedă o doză de creativitate. Ea este educabilă şi măsurabilă. Numeroşi psihologi susţin că toţi copiii sunt creativi până când adulţii le diminuează îndrăzneala, spontaneitatea şi originalitatea.

APTITUDINILE

Oamenii se deosebesc între ei după posibilităţile lor de acţiune. Ştim cu toţii că aceleaşi acţiuni, fie ele practice, artistice, intelectuale, sportive sau de altă natură, sunt executate de diverşi indivizi la diverse niveluri calitative, cu o eficienţă mai mare sau mai mică, uneori foarte redusă.

Reuşita în activităţi este datorată unor însuşiri de personalitate grupate sub numele de „aptitudini”, cuvânt derivat

din limba latină unde „aptus” înseamnă „potrivit pentru...”

Aptitudinea este o însuşire sau un complex de însuşiri psihice şi fizice care asigură succesul, reuşita într-o activitate (I. Radu) sau însuşirea psihică relativ stabilă care constituie condiţia majoră pentru efectuarea cu succes a unor forme de activitate (Al. Roşca).

Aptitudinile sunt subsisteme sau sisteme operaţionale, superior dezvoltate, care mijlocesc performanţe supramedii în activitate. Ele se demonstrează întotdeauna prin reuşită în activităţi. Orice activitate se efectuează cu ajutorul unor mijloace sau instrumente, fapt evident în cazul disciplinei informatică.

Aptitudinile constituie latura instrumentală şi executivă a personalităţii. Este o instrumentaţie psihică; uneori se spune despre inteligenţă că este tăioasă sau pătrunzătoare, se vorbeşte despre fineţea auzului, despre urzeala imaginaţiei, despre concentrarea şi distributivitatea atenţiei. Toate acestea sunt posibile datorită gradului de dezvoltare a unor funcţii sau a îmbinării operaţiilor.

Există aptitudini de grup. Una dintre ele se numeşte „factorul de dexteritate manuală”. Aptitudinea de a utiliza obiectele, de a mânui diverse instrumente sau unelte exprimă mai degrabă o caracteristică psihomotorie decât una cognitivă. Ea este importantă în orientarea şcolară şi profesională, fiind deci

necesară evaluarea ei de către profesor sau de către o persoană specializată (psiholog sau consilier şcolar).

Aptitudinea tehnică este o aptitudine complexă, specială ce rezultă din îmbinarea şi compensarea reciprocă a mai multor aptitudini simple cum ar fi: dexteritatea manuală, percepţia spaţială, gândirea tehnică, inteligenţa, sesizarea şi utilizarea principiilor mecanice; - „este o formă a inteligenţei practice, ca atare, nivelul ei depinde de puterea de combinare pe plan mintal."[1]

În constituirea aptitudinilor tehnice, un rol important îl are interesul copilului pentru astfel de activităţi, precum şi mediul său. „Astfel, un copil care a fost solicitat în mod repetat de către tatăl său să asiste sau chiar să ajute la repararea maşinii sau a diferitelor aparate din casă, ar putea prezenta ulterior aptitudini tehnice mai dezvoltate."[2]

Referitor la aptitudinile şcolare, sarcina noastră, a profesorilor, constă în primul rând în cunoaşterea individualizată a nivelului de dezvoltare al aptitudinilor elevilor, apoi în adaptarea conţinutului şi metodelor didactice, în funcţie de acestea asigurând astfel eficienţa demersului didactic.

Atât aptitudinile, cât şi talentele trebuie depistate şi mai

[1] L., Rusu, apud. Al. Roşca, op. cit., p. 91.
[2] A., Cosmovici, L., Iacob, *Psihologie şcolară*, Iaşi, Editura Polirom, 1998, p. 78.

ales trebuie create condiţii pentru exersarea şi cultivarea lor.

„Prin origine, ele sunt o zestre individuală, dar prin consecinţe şi rezultate, ele fac parte din bogăţia spirituală a unui popor.”[1]

DEPRINDERILE

Deprinderile pot fi definite ca fiind componente automatizate ale activităţii, conştient elaborate, consolidate prin exerciţiu, dar desfăşurate fără control conştient permanent.

Acele componente ale activităţii care se repetă frecvent, se exersează mult, ajung să se automatizeze şi devin astfel deprinderi.

O activitate nu poate fi integral automatizată pentru că este declanşată şi orientată conştient, supravegheată în ansamblul ei şi controlată în ceea ce priveşte elementele variabile şi noi. Fiind rezultatul învăţării, deprinderile nu sunt inconştiente şi autonome, ci în anumite faze subconştiente, putând fi uşor trecute sub controlul conştient. Deprinderile se desfăşoară involuntar pentru că sunt post-voluntare, dar ele pot fi reluate ca acţiuni intenţionate, reconstruite conştient.

Cine dispune de deprinderi are o anumită cursivitate în mişcări, operaţii, acţiuni, evitându-se astfel opririle, reîntoarcerile.

[1] Ion, Drăgan, *Psihologia pentru toţi*, Bucureşti, Editura Ştiinţifică, 1991, p. 58.

În aceste cazuri activităţile sunt desfăşurate foarte uşor, fără efort şi totodată eficient, operativitatea deprinderilor fiind maximă.

Deprinderile pot fi grupate după câteva criterii:

a) după gradul complexităţii – deprinderi simple şi complexe;

b) după natura proceselor psihice, în care are loc automatizarea – deprinderi senzoriale-perceptive, verbale, de gândire, motrice;

c) după tipul de activitate în care ele se integrează – deprinderi de joc, de învăţare, de conduită morală, de lucru cu diverse aplicaţii.

2.2. FACTORII CREATIVITĂŢII

Factorii care determină sau pot influenţa creativitatea sunt foarte numeroşi şi variaţi. Ei se pot combina în structuri foarte diferite, ceea ce face ca fiecare creator, chiar atunci când este vorba de acelaşi domeniu de activitate sau aproximativ acelaşi nivel al creaţiei, să aibă particularităţile sale caracteristice, personalitatea sa proprie, care-l diferenţiază de ceilalţi.

Factorii creativităţii pot fi grupaţi în: subiectivi (însuşiri ale persoanei, ale subiectului creator) şi obiectivi (condiţiile obiective ale vieţii, cu deosebire condiţiile social-educative). Factorii subiectivi se pot grupa în: factori intelectuali, aptitudinali şi factori nonintelectuali şi nonaptitudinali – mai ales factorii motivaţionali, temperamentali şi de caracter.

FACTORII INTELECTUALI

Studiul factorilor intelectuali ai creativităţii a mers în două direcţii:

a) aplicarea testelor de inteligenţă;
b) investigarea gândirii creatoare.

Potrivit cercetătorului J.P. Guilford gândirea creatoare este în esenţă o gândire divergentă. O astfel de gândire este orientată în direcţii diferite, spre o diversitate de soluţii, căci pot să existe o mulţime de soluţii, mai mult sau mai puţin adecvate, iar căi date de rezolvare nu sunt; ele trebuie descoperite prin varierea direcţiei gândirii, plecând de la o informaţie dată.

În cadrul gândirii divergente se pot distinge mai multe aspecte: fluiditatea, flexibilitatea, originalitatea.

După Guilford, fluiditatea poate fi asociativă, ideaţională şi de expresie, la care se adaugă fluiditatea cuvântului. Fluiditatea exprimă bogăţia şi uşurinţa actualizării asociaţiilor şi desfăşurarea uşoară a ideilor.

Fluiditatea nu poate fi o componentă a creativităţii, dacă elementele asociativ-verbale nu exprimă şi un conţinut de idei. Ea este implicată atât în gândirea reproductivă, cât şi în gândirea creatoare.

Flexibilitatea poete fi spontană sau adaptivă. Ea este

spontană atunci când sarcina dată subiectului nu conţine nimic care să-i sugereze să fie flexibil. În cazul flexibilităţii adaptive este vorba de o modificare în modul de a aborda sau interpreta o situaţie, în strategia utilizată, fără de care nu se poate ajunge la o rezolvare, la o soluţie valabilă.

După părerea lui Alexandru Roşca, flexibilitatea gândirii este principala componentă cognitivă a creativităţii. „Prin ea se înţelege restructurarea promptă şi adecvată a informaţiei, a sistemului de cunoştinţe în conformitate cu cerinţele noii situaţii, modificarea modului de abordare când cel anterior nu se dovedeşte eficient.”[1]

Flexibilitatea gândirii are ca o caracteristică esenţială restructurarea uşoară a vechilor legături temporare, a vechilor asociaţii, în conformitate cu cerinţele noii situaţii; o uşoară schimbare a punctului de vedere, a modului de abordare a unei situaţii sau probleme în funcţie de cerinţele noii situaţii, a direcţiei gândirii. Opusul flexibilităţii este inerţia sau rigiditatea gândirii, care înseamnă perseverarea într-o situaţie nouă cu modalităţi anterioare de rezolvare a problemelor, cu alte cuvinte greutatea sau imposibilitatea de a vedea o situaţie dintr-un unghi nou de vedere, manifestarea stereotipiei în gândire.

[1] Al., Roşca, *Creativitatea generală şi specifică, Bucureşti*, Editura Academiei, 1981, p. 55.

Flexibilitatea nu este incompatibilă cu perseverarea în căutarea unei soluţii. Renunţarea uşoară la realizarea unui obiectiv creativ, în faţa celor mai mici dificultăţi, ar însemna superficialitate şi instabilitate.

A treia componentă a creativităţii ar fi originalitatea. Ca indiciu al originalităţii ar fi caracterul neuzual al răspunsurilor şi soluţiilor, varietatea lor statistică. Cota de originalitate corespunde distanţei dintre produsul nou şi ceea ce preexistă ca fapt cunoscut şi uzual în domeniul respectiv.

Gândirea, prin însăşi natura ei, are un caracter creator. Gândirea creatoare are nevoie de un material bogat cu care să se opereze şi care să faciliteze generalizarea. Trebuie încurajată de către cadrul didactic, dobândirea de cunoştinţe bogate şi variate, cu care să opereze şi pe care să le aplice cât mai variat. O sărăcie de informaţii este un obstacol în formarea creativităţii gândirii. Îmbogăţirea cunoştinţelor nu înseamnă simpla lor însumare, ci însuşirea structurilor interioare, sistematizarea lor şi înţelegerea relaţiilor dintre ele. Gândirea nu operează în gol, ci pe baza informaţiei stocate prin procesul de memorare. Cunoştinţele însuşite mecanic sunt un obstacol pentru creativitatea creatoare a gândirii. O memorie bună, bine stocată şi organizată nu poate fi un neajuns; ea aduce o contribuţie indirectă, dar importantă la realizarea unor performanţe creatoare.

FACTORII APTITUDINALI

Deşi sunt foarte importanţi, factorii intelectuali, nu pot garanta singuri creativitatea şi nici nu pot asigura obţinerea unor performanţe umane. Este necesară prezenţa unor aptitudini speciale, pe lângă implicaţia unor factori motivaţionali sau a unor însuşiri şi trăsături de personalitate. Sunt persoane cu înzestrare intelectuală superioară, dar care în ceea ce priveşte aptitudinile speciale (pentru desen, pictură, muzică, sport, literatură, matematică, etc.) sunt la medie sau chiar sub medie, după cum sunt persoane care prezintă una sau mai multe aptitudini într-un grad dezvoltat, iar în ceea ce priveşte inteligenţa sunt de nivel mediu sau chiar sub medie.

Aptitudinile speciale nu sunt aspecte ale intelectului. Corelaţia lor cu inteligenţa este ridicată, dar, obişnuit, această corelaţie este redusă sau chiar inexistentă. Aptitudinile sunt însuşiri psihice şi fizice relativ stabile care conferă activităţilor eficienţă, fiind suportul psihologic important al randamentului înalt şi al realizării cu succes a unor activităţi. Este caracteristic faptul că persoanele care posedă aptitudini dau rezultate remarcabile ca urmare a unor momente de educaţie, faţă de alţii care la aceeaşi cantitate şi calitate de învăţare au rezultate mai puţin strălucite.

Important în problema aptitudinilor, mai ales a celor

artistice, este că au la bază componente complexe senzoriale şi abilităţi manuale, dar şi o atracţie deosebită faţă de activităţile în cauză.

Se poate considera că nu există persoană normală care să nu aibă nici un fel de aptitudini. Mai frecvent este cazul în care acestea nu se conştientizează de către cel ce le posedă.

Aptitudinea generală nu corelează cu inteligenţa. Ca un copil să poată fi considerat superior înzestrat pentru muzică trebuie să obţină o cotă ridicată la toate testele care măsoară diferite aspecte ale aptitudinii muzicale. Lipsa unei corelaţii semnificative între inteligenţă şi aptitudinea muzicală s-ar explica prin faptul că această aptitudine implică, pe lângă anumiţi factori (inteligenţă, imaginaţie, etc.) şi existenţa unor calităţi psihofiziologice, cum sunt discriminarea auditivă şi anumite funcţii motorii, toate acestea necorelând cu inteligenţa.

Aptitudinile pentru desen corelează foarte puţin sau deloc cu inteligenţa generală. Acest lucru se explică prin prezenţa unor factori psihofiziologici care sunt implicaţi în aptitudinea picturală, de exemplu: acuitatea vizuală, abilitatea manuală.

Foarte complex, implicând numeroase aptitudini, este talentul literar. Tolstoi spunea că în scriitor trebuie să se întâlnească trei oameni: gânditorul, artistul şi criticul. Important pentru scriitor este nu numai să ştie să observe, ci şi să elaboreze

observaţiile, să le combine în imagini vii şi să le găsească exprimarea verbală adecvată.

Aptitudinea tehnică, manifestată îndeosebi în cadrul disciplinei TIC corelează foarte puţin sau deloc cu inteligenţa. Această aptitudine se structurează în diferite componente cum ar fi: percepţia spaţială, imaginaţia, gândirea creativă, dexteritatea manuală, ingeniozitatea şi capacitatea de a utiliza diferitele aplicaţii folosite în realizarea unor produse.

Caracterul diferenţiat al aptitudinilor specifice diferitelor sectoare de manifestare rezultă atât din varietatea aptitudinilor simple care intră în sistem, cât şi din diferenţele de intensitate şi calităţi operaţionale ale acestor componente.

Persoanele dotate pentru anumite activităţi vor prezenta aranjamente strict personale ale verigilor intelectuale şi ale aptitudinilor speciale. „Toate procesele psihice sunt implicate în evoluţia creatoare, dar problema principală este aceea a modului în care ele sunt corelate şi orientate, a modului în care sistemul devine emergent.”[1]

FACTORII NONINTELECTUALI ŞI NONAPTITUDINALI

O persoană nu va atinge niciodată nivelurile de care ar fi în

[1] Paul, Popescu-Neveanu, *Dicţionar de psihologie*, Bucureşti, Editura Albatros, 1978, p. 153.

stare, dacă aptitudinile sale nu sunt dinamizate şi susţinute la un tonus corespunzător, de o curiozitate vie, de dorinţa de a realiza ceva, de tenacitate. Anumite trăsături de personalitate, cum ar fi sistemul de răspundere, capacitatea de efort prelungit, conştiinţa responsabilităţii faţă de om, de materialul pe care-l lucrezi, de produsul realizat, au funcţie de catalizatori pentru factorii intelectuali.

Ele se constituie în condiţii predictive ale reuşitei creatoare şi totodată menţin potenţialul creativ.

Studiind raportul inteligenţă-creativitate-personalitate, Paul Popescu-Neveanu ajunge la concluzia că dincolo de un nivel mediu al inteligenţei, rolul decisiv în determinarea performanţei creative îl deţin factorii de personalitate numiţi sintetic „aptitudine creativă”.

Motivaţia este considerată o componentă esenţială a creativităţii. O motivaţie puternică (curiozitate, interes) constituie un factor foarte important în extinderea performanţelor ştiinţifice pe o perioadă de timp mai îndelungată.

Pentru că în calea oricărei activităţi creatoare pot apărea obstacole numeroase, o atitudine activă în faţa dificultăţilor este o trăsătură de caracter de cea mai mare importanţă în activitatea de creaţie.

O altă caracteristică a personalităţii creatoare este

atitudinea faţă de muncă. O muncă tenace caracterizează pe mulţi artişti şi oameni de ştiinţă. Atât la cercetători, cât şi la artişti, creativitatea nu vine de la o inspiraţie care ar invada brusc o minte inertă, ci de la munca unei persoane active.

Un rol important revine în procesul de creaţie unor factori cum sunt: sensibilitatea faţă de mediul său, iniţiativa, încrederea în posibilităţile proprii şi ataşamentul faţă de munca sa, îndrăzneala în gândire.

În schimb indecizia, ezitarea, timiditatea excesivă, frica de critică sau eşec, autodescurajarea sunt factori ce pot inhiba activitatea de creaţie.

FACTORII SOCIALI

Nici o altă invenţie sau descoperire n-a apărut din nimic, fără vreo legătură cu alte invenţii sau descoperiri similare.

Întreaga noastră imaginaţie creatoare nu este produsul unui creier uman izolat, ci al unui creier care a fost condiţionat de interacţiunea cu alţi oameni, de întreaga istorie a civilizaţiei.

Personalitatea creatoare, în oricare domeniu de activitate, nu este în afara contextului social în care trăieşte şi creează.

Gândirea fiecărui om se formează şi se îmbogăţeşte prin contactul şi confruntarea cu ideile altor oameni, în procesul asimilării creatoare a cunoştinţelor generate de contemporani şi

înaintaşi. Faptul că în istoria ştiinţei se întâlnesc frecvent cazuri de descoperiri simultane ale unor creatori care au lucrat independent, constituie o dovadă a rolului factorilor sociali.

Rolul factorilor sociali, în general a condiţiilor de viaţă şi muncă ale omului, este foarte important în activitatea de creaţie, în orice domeniu. Oricare ar fi domeniul de activitate, personalitatea creatoare nu există în afara contextului social în care ea trăieşte şi creează.

J. Piaget afirmă că: „societatea este unitatea supremă, iar individul nu ajunge la invenţiile şi construcţiile sale intelectuale decât în măsura în care el este sediul de interacţiuni colective, ale căror nivel şi valoare depind natural de ansamblul societăţii. Marele om care pare să lanseze curente noi nu este decât un punct de intersecţie sau de sinteză a unor idei elaborate printr-o cooperare continuă."

Un rol important îl au condiţiile mediului familial, influenţele procesului de învăţământ, ale prietenilor, ale colectivului de muncă.

Nici unul din factorii amintiţi, luat izolat, nu poate să asigure performanţa creatoare.

Performanţele creatoare superioare sunt rezultatul combinării fericite a factorilor ce favorizează creativitatea, combinare care diferă de la individ la individ. Alexandru Roşca

afirmă că „factorii creativităţii sunt totdeauna implicaţi în performanţele creatoare superioare, ei se prezintă la nivele diferite şi sub aspecte calitativ diferite, putându-se în acelaşi timp, compensa reciproc (un factor mai proeminent poate compensa prezenţa la un nivel mai puţin ridicat al altui factor)”.[1]

[1] Al., Roşca, *Creativitatea*, Editura Orizonturi, Bucureşti, 1972, p. 73

2.3. GÂNDIREA CRITICĂ – UN ALT FEL DE ÎNVĂŢARE

A gândi critic înseamnă a fi curios, a pune întrebări, a căuta răspunsuri, a căuta cauze şi implicaţii, a găsi alternative la atitudini deja fixate, a adopta o poziţie pe baza unei întemeieri argumentate şi a analiza logic argumentele celorlalţi. Este un proces activ care îl face pe cel care învaţă să deţină controlul asupra informaţiei, interogând-o, reconfigurând-o, adaptând-o sau respingând-o. Este evident că o asemenea capacitate nu se dezvoltă de la sine, ci ea trebuie exersată şi încurajată într-un mediu de învăţare adecvat.

Gândirea critică este una dintre abilităţile cheie care asigură succesul în societatea cunoaşterii şi comunicării.

Gândirea ca proces psihic, într-o accepţiune generală, reprezintă capacitatea specific umană de a introduce ordine, raţionalitate în evenimentele lumii. Aşadar, cu ajutorul gândirii ordonăm, clasificăm, comparăm, categorisim.

Termenul critic se referă, mai ales din perspectivă constructivistă, nu la critica ce distruge, ci la cea care construieşte.

A gândi critic constructiv, arată I. Al. Dumitru, înseamnă "a susţine cu argumente convingătoare, raţionale, anumite opinii şi ale respinge pe altele, a te "îndoi" cu scopul de a obţine noi argumente care să-ţi întărească sau, dimpotrivă, să-ţi şubrezească

propriile convingeri şi credinţe, a supune analizei şi evaluării orice idee personală sau aparţinând altora”[1]

Caracteristicile gândirii critice:

- Presupune formularea de către fiecare elev a unor păreri proprii, personale, eventual originale referitoare la o problemă;
- Dezbaterea responsabilă a ideilor şi soluţiilor avansate de fiecare individ, în mod individual sau în grup;
- Alegerea raţională a unei soluţii optime dintre mai multe posibile;
- Rezolvarea de probleme în timp optim şi cu eficienţa scontată.

Gândirea critică are două planuri esenţiale:

1. social: potrivit căreia învăţarea şi munca în colaborare determină construirea şi manifestarea solidarităţii umane.

2. pragmatic: învăţarea bazată pe dezvoltarea gândirii critice creează posibilitatea implicării plenare a elevilor în activitate, pornindu-se de la stârnirea curiozităţii acestora şi continuând cu implicarea efectivă a lor în rezolvarea unor probleme autentice, de viaţă.

[1] Dumitru, I. A., *Dezvoltarea gândirii critice şi învăţarea eficientă*, Editura de Vest, Timişoara, 2000, p.25-26

Dimensiuni pe care le dezvoltă gândirea critică

Pe măsură ce îşi dezvoltă capacitatea de a gândi critic, elevii progresează pe următoarele patru planuri:

1. De la personal (a gândi pentru tine) la public (a gândi şi pentru ceilalţi).

Primele reacţii ale copiilor referitoare la un anumit lucru sunt, de obicei, exprimate sub forma "îmi place"/"nu-mi place", cu alte cuvinte sunt reacţii la ceea ce ei, ca persoane, găsesc interesant. Pe măsură ce se maturizează şi acumulează experienţă, elevii îşi dezvoltă capacitatea de a-şi exprima reacţiile în termeni care pot fi înţeleşi de alţii şi care sunt mai adecvaţi pentru comparaţii şi dezbateri. O raportare personală la o idee, aşa cum arată James Britton, este o sursă de vitalitate şi autenticitate în gândire. Cu toate acestea, semnul distinctiv al unei persoane educate este capacitatea sa de a-şi exprima gândurile clar şi convingător în faţa celorlalţi, fie ei cunoscuţi sau străini. Aceasta presupune, de fapt, trecerea de la reacţia primară personală, la exprimarea ei nuanţată, într-o manieră adecvată, în public. Această reacţie este rodul experienţei şi educaţiei persoanei.

2. De la heteronom(atribuirea înţelepciunii şi autorităţii celorlalţi), la autonom (siguranţa cunoştinţelor şi posibilităţilor proprii de gândire).

Aceasta presupune trecerea de la credinţa că “ceea ce este

scris este adevărat", la posibilitatea de a te îndoi de adevărurile scrise ale unor personalităţi şi de a aduce şi alte adevăruri, chiar proprii, justificându-le şi argumentându-le. Cu alte cuvinte se face trecerea spre o gândire autonomă, flexibilă, critică. Elevii care îşi dezvoltă autonomia de gândire devin mai siguri pe ei înşişi, mai dispuşi să adopte un punct de vedere şi să-l susţină şi mai înclinaţi să pună sub semnul întrebării validitatea unui argument exprimat într-un text.

3. De la intuitiv (experenţial, limitat), la logic (formalizat, comunicabil).

A afirma lucruri în mod intuitiv înseamnă a exprima idei fără a reflecta prea mult la legătura dintre afirmaţie şi experienţă. A te concentra asupra logicii înseamnă a deveni sensibil la modul în care sunt aranjate probele pentru a susţine concluzia. Logica nu ameninţă să înlocuiască în totalitate intuiţia dar, în măsura în care logica poate fi o formă mai publică de gândire decât intuiţia, democraţia are de câştigat atunci când oamenii sunt capabili să-şi expună poziţiile în mod logic. Fără a nega rolul intuiţiei, trebuie să remarcăm că logica presupune un nivel profund al rezolvării tuturor problemelor cu care ne confruntăm. Mai ales pentru formarea gândirii critice, pentru prezentarea şi susţinerea argumentelor, trebuie să facem apel la logică.

4. De la o perspectivă (închiderea în propriile convingeri),

la perspective multiple (capacitatea de a ţine cont şi de părerile celorlalţi).

Un gânditor mai puţin matur se va crampona în propriile convingeri, indiferent de ceea ce spun ceilalţi. Un gânditor mai matur, un gânditor critic va ţine cont şi de convingerile celorlalţi. Gânditorul mai matur poate, cu siguranţă, să-şi modifice convingerile când ajunge să fie convins de argumentele altuia; dar la fel de important este faptul că, susţinându-şi propria poziţie, este capabil să accepte şi poziţiile altora, exprimând acest lucru în maniera: "Ştiu că sunt mulţi care cred X, dar daţi-mi voie să vă arăt motivele pentru care Y e preferabil". Un adevărat gânditor critic este cel care acceptă mai multe perspective de abordare a unei probleme, conştientizând avantajele şi dezavantajele fiecărei perspective.

Dintre toate caracteristicile de mai sus, observăm faptul că abilitatea de a gândi critic presupune o anumită experienţă de viaţă, un antrenament special în abordarea perspectivelor multiple, deschiderea spre nou şi spre împărtăşirea experienţelor proprii celorlalţi membri ai grupului.

Gândirea critică poate fi privită în triplă ipostază: ca strategie de gândire, ca investigaţie şi ca proces.

a) Ca strategie de gândire, gândirea critică este văzută ca o strategie ce apelează la mai multe operaţii coordonate ale gândirii.

Într-un mod asemănător, gândirea critică este înţeleasă de unii autori români ca „un mod de a aborda şi de a rezolva problemele, bazat pe argumente convingătoare, logic-raţionale”.

b) Ca investigaţie, gândirea critică reprezintă „o cercetare al cărei scop este explorarea unei situaţii, unui fenomen, unei întrebări sau unei probleme, cu scopul de a formula o ipoteză sau o concluzie care să reunească toate informaţiile disponibile şi să fie demonstrabilă într-o manieră convingătoare”.[1]. Aşadar, gândirea critică poate fi înţeleasă ca o investigaţie care să conducă la o concluzie justificată.

c) Ca proces, gândirea critică parcurge o succesiune de etape:

•apariţia problemei - punerea în acţiune a atitudinilor şi capacităţilor necesare - rezolvarea problemei; sau:

•apariţia unei situaţii neaşteptate - evaluarea situaţiei - căutarea explicaţiilor sau a soluţiilor - construirea unor perspective diferite - rezolvarea situaţiei.

Pe lângă desfăşurarea în etape, gândirea critică se caracterizează şi printr-o dezvoltare în stadii, începând cu certitudini absolute, trecând prin îndoială şi sfârşind cu înţelegerea faptului că există mai mult de o singură abordare şi mai multe

[1] Steele, J. L., Meredith, K.S., Temple, Ch., (1998), Un cadru pentru dezvoltarea gândirii critice la diverse materii de studiu, Ghidul I, p.1109

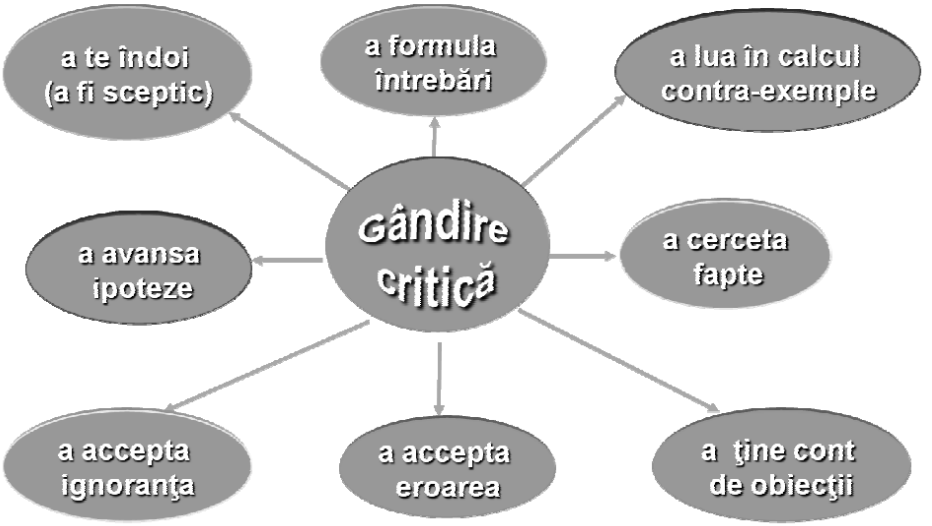
soluţii ale oricărei probleme.

Gândirea critică presupune exercitarea unor operaţii de gândire pentru rezolvarea unei probleme sau, mai exact, a unei situaţii-problemă. Aşadar, a gândi critic înseamnă a mobiliza acele operaţii intelectuale necesare pentru rezolvarea unei anumite probleme.

Scopul gândirii critice este, deci, să ajungă la înţelegere, la evaluarea punctelor de vedere şi să rezolve probleme. Cum toate aceste trei domenii implică punerea de întrebări, putem spune că gândirea critică este chestionarea sau cercetarea în care ne angajăm atunci când căutăm să înţelegem, să evaluăm sau să rezolvăm[1].

[1] Maiorana, Victor P. Critical Thinking across the Curriculum: Building the Analytical Classroom,1992

Gândirea critică - abilitate cognitivă superioară



Utilizarea competenţelor TIC în cadrul disciplinelor informatice a avut şi va avea în continuare în vedere stimularea şi intensificarea comunicării pe toate palierele, dezvoltarea gândirii critice, înţelegerii realităţii şi a contribuit în mod esenţial la dezvoltarea competenţelor generale pentru informatică.

Elevii trebuie să înveţe să evite gândirea mecanică, emotivă şi resentimentul faţă de oponenţi. Gândirea critică permite elevilor să gândească raţional decât sentimental şi să aducă argumente logice şi nu false.

Iniţierea în gândirea critică este preferabil să se facă de la vârste fragede. Gândirea critică ajută la dezvoltarea unei analize bune şi abilităţilor de rezolvare a problemelor, iar gândirea creativă contribuie la creşterea performanţelor profesionale. Utilizarea metodelor de gândire critică şi creativă în procesul educaţional la diverse discipline şi activităţi extracurriculare ajută elevii să obţină performanţe mai bune la materii şi să crească capacitatea lor de a evalua calitatea informaţiilor prezentate de profesori şi însuşite de ei.

2.4. PARTICULARITĂŢI DE DEZVOLTARE PSIHOLOGICĂ A ŞCOLARULUI MIC

DEZVOLTAREA COGNITIVĂ

Dezvoltarea psihocognitivă a şcolarului mic reprezintă saltul de la gândirea de tip funcţional la gândirea de tip categorial.

Comparativ cu alte perioade, în mica şcolaritate transformările psihice se fac lent şi nespectacular, dar sunt fundamentale pentru evoluţia ulterioară a copilului.

După Piaget, stadiile dezvoltării intelectuale sunt următoarele: perioada senzorio-motorie (0-2 ani), perioada preoperaţională (2-7 ani), perioada operaţiilor concrete (7-11 ani) şi perioada operaţiilor formale (11 ani- maturitate).

Gândirea

Şcolarul mic se află, după vârsta pe care o are, în a treia perioadă de dezvoltare cognitivă, perioada operaţiilor concrete. În această perioadă, gândirea unui copil este foarte asemănătoare cu aceea a unui adult, dar copilul are totuşi dificultăţi cu noţiunile abstracte, pentru că, pentru a le înţelege, trebuie să le lege de lumea reală. Copiii aflaţi în această perioadă sunt caracterizaţi de o dorinţă extraordinară de a culege informaţii despre lume. Deseori, ei adună liste considerabile de fapte sau de date despre un subiect de interes.

În cadrul acestei etape de viaţă încep să se contureze diferenţele în baza variabilei sex la nivelul unor abilităţi speciale şi ale randamentului şcolar. Fetele pot prezenta performanţe superioare în ceea ce priveşte fluenţa verbală, ortografia, citirea şi calculul matematic. Băieţii au performanţe mari la raţionamentul matematic, orientarea spaţială, precum şi la soluţionarea unor probleme de descoperire.

Cu mai mare claritate încep să se precizeze în această perioadă şcolară diferenţa dintre stilurile cognitive. Acest concept desemnează tendinţa de a răspunde varietăţii sarcinilor şi problemelor intelectuale într-un mod particular. Cercetările care vizează stilul cognitiv fac deosebirea între stilul impulsiv faţă de stilul reflexiv şi stilul analitic faţă de cel tematic.

Copiii impulsivi au un ritm de conceptualizare rapid, cu tendinţa de a ieşi „la rampă" cu primul răspuns care le vine în minte şi sunt preocupaţi să găsească repede răspunsuri. Copiii reflexivi au nevoie de timp înainte de a răspunde; ei par a valoriza posibilitatea de analiză a variantelor de răspuns, fiind preocupaţi mai ales de calitatea răspunsului şi nu de rapiditatea cu care îl oferă.

În ceea ce priveşte copiii cu stil cognitiv analitic, ei pleacă de la conceptualizare la detalii, faţă de cei cu stil tematic, care iau în considerare întregul.

Din punct de vedere al randamentului acestor stiluri cognitive, copiii impulsivi dau rezultate mai bune în sarcini care solicită interpretări globale. Cei reflexivi au performanţe mai mari în sarcini de tip analitic. Trebuie reţinut faptul că problema stilului nu se pune în termeni de superioritate sau inferioritate.

Procesele senzoriale

Pe plan perceptiv au loc în perioada şcolară mică o serie de achiziţii. Se dezvoltă sensibilitatea discriminativă şi pragurile perceptive absolute. Tot acum se conturează evaluări din ce în ce mai fine legate de mărime, greutate, perceperea structurii materialelor. Totuşi, evaluarea mărimii este încă deficitară (copiii de 6- 7 ani supraestimează mărimile, iar cei de 8- 9 ani subestimează mărimile şi distanţele). Se dezvoltă importante aspecte discriminative în legătură cu spaţiul mic: orientarea spaţială pe foaia de hârtie, orientarea dreapta – stânga, sus–jos, în rândurile orizontale ale scrierii (această activitate cuprinde antrenarea memoriei, a inteligenţei, a atenţiei, a reprezentărilor pornind de la percepţie). Spaţiul se organizează acum şi ca distanţă psiho-afectivă sub trei dimensiuni: spaţiul intim, spaţiul personal şi spaţiul oficial. Spaţiul posedă mărime, formă, extensivitate, detalii şi relaţii personale.

În privinţa percepţiei timpului se remarcă faptul că timpul subiectiv începe să se raporteze la timpul cronometrabil, care

capătă consistenţă. Ceasul şi citirea lui devine instrument al autonomiei psihice. Anul începe să fie considerat de 365 zile, cu patru anotimpuri,etc., iar evenimentele încep să se raporteze la aceste repere. (Şchiopu, U., 1995)

Atenţia

Pe la 6-7 ani, în mod normal, copilul este capabil de o atenţie suficient de stabilă pentru a se putea integra în activitatea şcolară. Totuşi, profesorul are încă de luptat, mai ales în primul an de şcoală, cu unele neajunsuri ale atenţiei elevilor.

O altă caracteristică a atenţiei copilului de vârstă şcolară mică este predominarea atenţiei involuntare asupra celei voluntare. Din această cauză, dacă lecţia nu suscită suficient interes, dacă nu declanşează stări afective pozitive, copiii devin repede neatenţi.

Treptat, copilului i se formează deprinderea de a fi atent în diverse situaţii, în raport cu cerinţele interne şi externe; implicit se dezvoltă diferitele particularităţi ale atenţiei voluntare, care se împletesc cu aspectele pozitive ale atenţiei involuntare.

Memoria

La vârsta şcolară mică sporeşte treptat ponderea memoriei voluntare şi a memoriei logice. În activitatea şcolară trebuie să se exerseze sistematic procesele memoriei, să se îmbogăţească volumul acesteia, să se dezvolte rapiditatea întipăririi, trăinicia

păstrării şi fidelitatea actualizării.

Copiii de vârstă şcolară mică manifestă tendinţa de a memora textual cunoştinţele predate fără a le înţelege. Deci, micii şcolari fac uz mai frecvent de memoria mecanică decât de cea logică.

Cunoscând particularităţile memoriei la vârsta şcolară mică, profesorul trebuie să se preocupe de dezvoltarea caracterului intenţional şi logic al memoriei, încă din primele clase.

La şcolarul mic toate aceste procese cognitive (gândire, atenţie, memorie) sunt stimulate cel mai pregnant în cadrul activităţii de învăţare care devine predominantă.

Este lesne de înţeles, deci, că odată cu intrarea în şcoală, copilul începe să îndeplinească o activitate serioasă, apreciată ca atare de familie şi de membrii societăţii. Prin activitatea principală desfăşurată acum – învăţătura – i se dezvoltă atât dimensiunea intelectuală, cât şi cea morală, i se formează aspectele structurale ale personalităţii.

DEZVOLTAREA PSIHO-MOTRICĂ

Motricitatea este una din manifestările care exprimă foarte multe achiziţii făcute de copil în acest stadiu. Motricitatea este şi terenul pe care se manifestă cele dintâi reglaje voluntare, pentru că scopurile atinse prin acţiuni reale sunt cele mai accesibile copiilor.

Este binecunoscut faptul că, dacă un copil este bine

dezvoltat din punct de vedere psiho-motric, poate ajunge la o înaltă dezvoltare cognitivă, poate face faţă cu mai mare uşurinţă cerinţelor impuse de viaţa de şcolar.

Disciplina opţională „Prietenul meu, calculatorul” are un rol important în dezvoltarea intelectuală a şcolarilor mici. Procesul activităţii practice la această vârstă creează largi posibilităţi pentru fixarea, sistematizarea şi verificarea cunoştinţelor însuşite la alte discipline.

Deprinderile de utilizare a celor mai simple aplicaţii uşurează, mai târziu, înţelegerea construcţiei şi funcţionării unor maşini complexe, moderne. Este de reţinut şi faptul că, la această vârstă, apare o atitudine nouă a copiilor, aceea de a-şi perfecţiona mişcările şi de a executa cu plăcere şi cu o mai mare responsabilitate aplicaţii practice la calculator.

Punctul de plecare în formarea deprinderii senzorio-motorii îl constituie de regulă perceperea unui model extern al actului, însoţit de explicaţii verbale pentru a-l desluşi mai bine. Modelul extern nu operează direct asupra musculaturii efectoare a celui care învaţă. Influenţa acestui model devine efectivă prin intermediul unui model intern, care se formează treptat în capul subiectului, în cadrul unei prime faze cognitive, de orientare şi de familiarizare cu noua acţiune. Pe măsură ce activitatea este urmărită de repetate ori împreună cu lămuririle necesare, imaginea

iniţial aproximativă se transformă într-o reprezentare precisă ce capătă treptat expresie chinestezică. Are loc „transferul excitaţiei de la celula vizuală la cea chinestezică.”(I.P.Pavlov)[1] Acest model reglează mişcarea, direcţia, intensitatea ei, cu alte cuvinte, realizarea motorie. Desigur, numai lucrând efectiv cu elevii de vârstă şcolară mică aceştia dobândesc deprinderi, adică mişcări care se execută în mod automat. Elevul preia, prin intermediul percepţiei şi al comunicării verbale, momente şi aspecte care se dezvăluie nemijlocit observaţiei externe, apoi, gestul motor ca atare, modul de articulare şi topografia sa într-un ansamblu.

Sunt necesare o serie de tatonări, încercări şi erori prin care se aproximează actul respectiv, micul şcolar apropiindu-se tot mai mult de model. În acest timp creşte rapiditatea execuţiei.

Şcolarul trebuie să refacă pe cont propriu acest itinerar, până va ajunge la controlul proprioceptiv al actului motor. Toate aceste lucruri ne duc cu gândul la faptul că psihomotricitatea nu se reduce la activitatea motorie, ci ea implică şi manifestări ale funcţiilor perceptive şi intelectuale.

La vârsta şcolară mică, unele studii arată că băieţii au o dezvoltare motrică mai bună. Această motivare a dezvoltării motrice se bazează pe faptul că băieţii dispun de mai multă

[1] Ion, Radu, *Psihologie şcolară*, Bucureşti, Editura Ştiinţifică, 1974, p. 50.

energie, mişcările lor sunt mai rapide şi mai dinamice, ceea ce îi determină să prefere jocuri şi activităţi cu multă mişcare. Începând de la vârsta de 9-10 ani fetele încep să manifeste conduite tot mai agile şi mai fine. Această dezvoltare a motricităţii în cazul fetelor se datorează activităţilor practice efectuate la diverse ore, activităţi ce solicită intens agilitatea motorie, precum şi a interesului sporit pentru frumos şi pentru aspectul estetic de care dau dovadă micile şcolăriţe.

Prin activitatea de bază desfăşurată acum – învăţarea, în toate formele ei – i se dezvoltă şcolarului mic atât dimensiunea cognitivă, cât şi cea psiho-motrică şi se pun bazele dezvoltării personalităţii în ansamblul ei.

BIBLIOGRAFIE

[1] G. Albeanu, *Sisteme de operare,* Editura Petrion, Bucureşti, 1999.

[2] L. Arici, *Caiet de lucru pentru clasa a V-a,* Editura Polirom, 2003.

[3] L. Arici, *Caiet de lucru pentru clasa a VI-a,* Editura Polirom, 2003.

[4] A. Atanasiu, *Concursuri de informatică,* Editura Petrion, 1995.

[5] A. Atanasiu, *Cum se scrie un algoritm? Simplu!*, Editura Agni, Bucureşti, 1993.

[6] A. Atanasiu, R. Pintea, *Culegere de probleme Pascal,* Editura Petrion, Bucureşti, 1996.

[7] C. Bălcău, M. Boloşteanu, L. Deaconu, *On some extension of a classic game,* MITRE-2009, Chişinău.

[8] F. C. Buruiană, *Teoria jocurilor,* infoarena (http://infoarena.ro/teoria-jocurilor).

[9] E. Cerchez, M. Şerban, *Sisteme de calcul,* Ed. L&S Informat, Bucureşti, 1998.

[10] E. Cerchez, M. Şerban, *Informatica pentru gimnaziu,* Editura Polirom, 2002.

[11] A. Cosmovici, L. Iacob, *Psihologie şcolară*, Iaşi, Editura Polirom, 1998.

[12] B. Diamandi, *Informatica în 27 de lecţii,* Editura Petrion, Bucureşti, 1999.

[13] I. Diamandi, *Cum să realizăm jocuri pe calculator*, Editura Agni, Bucureşti, 1993.

[14] I. Drăgan, *Psihologia pentru toţi*, Bucureşti, Editura Ştiinţifică, 1991.

[15] I. A. Dumitru, *Dezvoltarea gândirii critice şi învăţarea eficientă*, Editura de Vest, Timişoara, 2000.

[16] L. Gârlea, A. Negreanu Maior, A. Pintea, *Informatică pentru grupele de performanţă Gimnaziu*, Editura Dacia, 2003.

[17] *http://forum.softpedia.com/topic/83805-sodoku-in-c/* .

[18] *http://olimpiada.info.*

[19] *http://ro.wikipedia.org/wiki/Sudoku.*

[20] Maiorana, P. Victor, *Critical Thinking across the Curriculum: Building the Analytical Classroom*, 1992.

[21] R. Mârşan, *Sisteme de calcul*, Ed. Didactică şi pedagogică, R.A.,Bucureşti, 1997.

[22] N. C. Matei, *Probleme de psihologie şcolară*, Bucureşti, Editura Didactică şi Pedagogică, 1978.

[23] M. Miloşescu, *Tehnologia Informaţiei,* Editura Teora, Bucureşti, 1999.

[24] P. Popescu-Neveanu, *Dicţionar de psihologie*, Bucureşti, Editura Albatros, 1978.

[25] P. Popescu-Neveanu şi colaboratorii, *Psihologie*, Bucureşti, Editura Didactică şi Pedagogică, 1993.

[26] I. Radu, *Psihologie şcolară*, Bucureşti, Editura Ştiinţifică, 1974.

[27] Al. Roşca, *Creativitatea generală şi specifică*, Bucureşti, Editura Academiei, 1981.

[28] Al. Roşca, *Creativitatea*, Bucureşti, Editura Enciclopedică Română, 1972.

[29] J. L. Steele, K. S. Meredith, Ch. Temple, *Un cadru pentru dezvoltarea gândirii critice la diverse materii de studiu, Ghidul I*, 1998.

[30] U. Şchiopu, *Creativitatea potenţială şi virtuală*, Revista de Psihologie Nr. 3, 1979.

[31] A. Tauru, *Rolul şi locul abilităţilor practice în orientarea socială a copiilor*, „Învăţământul primar", Editura Discipol, Nr. 1/ 2001.

[32] S. Tudor, *Turbo Pascal pentru cei mici*, Editura L&S Informat, 1999.