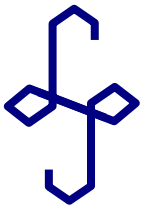
LICEUL DE INFORMATICĂ “GRIGORE MOISIL” IAŞI



REVINFO

Nr.2, Anul I

REDACŢIA REVISTEI

Coordonatori::

prof Oana Cristina Butnăraşu
prof. Ingrid Simona Iuscinschi

Colaboratori :

Profesori :

prof. Emanuela Cerchez
prof. Silvia Grecu
prof. Cornelia Ivaşc
prof. Marinel Şerban

Elevi :

Bogdan Locovei
Mădălina Melinte
Alexandru Paicu
Cristian Tudorache
Răzvan Ursachi

Editori :

Maricica Mera
Mădălina Melinte

CUPRINS

Lotul național de informatică se pregăteşte la Liceul de Informatică "Grigore Moisil" Iaşi

În perioada 14-21 iunie 2008 se va desfăşura la Liceul de Informatică "Grigore Moisil" Iaşi tabăra de pregătire a lotului național de informatică. Elevii vor participa la cursuri de pregătire şi vor susține baraje de selecție pentru stabilirea echipelor reprezentative ale României la Olimpiada Internațională de Informatică, Egipt 16-23 august 2008 (http://www.ioi2008.org) şi respectiv Balcaniada de Informatică pentru juniori, Bulgaria, 8-13 iulie 2008 (http://jboi2008.com).

Lotul național de informatică (seniori):

Nr.	Nume	Prenume	Liceul	Judet
1.	Tătăroiu	Bogdan-Cristian	Liceul Internațional de Informatică	Bucureşti
2.	Gheorghe	Cosmin	Liceul Internațional de Informatică	Bucureşti
3.	Filip	Stefan-Alexandru	C. N. de Informatică "T. Vianu"	Bucureşti
4.	Băltescu	Paul-Dan	Liceul Internațional de Informatică	Bucureşti
5.	Buruiană	Filip	C. N. "V. Alecsandri"	Galați
6.	Țandrău	Alexandru-Octavian	Liceul Internațional de Informatică	Bucureşti
7.	Rusu	Victor	Liceul Internațional de Informatică	Bucureşti
8.	Simion	Alexandru-Ciprian	Liceul Internațional de Informatică	Bucureşti
9.	Duță	Vlad	C.N. "Mircea cel Bătrân"	Vâlcea
10.	Tudose	Vlad	Liceul de Informatică "Gr. Moisil"	Iaşi
11.	Crețu	Iulian	C. N. "Ştefan cel Mare"	Suceava
12.	Schneider	Ştefan	Liceul Teoretic "A. Muller Guttenbrunn"	Arad

13.	Hevele	Istvan	Liceul Teoretic "Orban Balazs"	Harghita
14.	Ionescu	Victor-Cristian	C.N. "I. L. Caragiale"	Prahova
15.	Dima	Mircea	C.N. "Gheorghe Vrânceanu"	Bacău
16.	Horlescu	Marina	C.N. "Gheorghe Vrânceanu"	Bacău

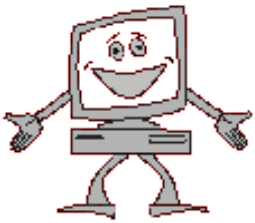
Lotul național de informatică (juniori):

Nr	Nume	Liceu	Județ
1	Gavrilă Vlad-Alexandru	Liceul Internațional de Informatică	Bucureşti
2	Stan Serban-Andrei	Liceul Internațional de Informatică	Bucureşti
3	Voroneanu Radu	C.N. "Mihai Viteazul" Ploiesti	Prahova
4	Taloi Bogdan-Cristian	Liceul Internațional de Informatică	Bucureşti
5	Purice Andrei	C. N. "I. L. Caragiale" Ploieşti	Prahova
6	Tamaş Iulia	C.N. "Mihai Viteazul" Ploiesti	Prahova
7	Mateescu Maria	C. N. "Emil Racoviță"	Iaşi
8	Zernoveanu Radu	Liceul Internațional de Informatică	Bucureşti
9	Pripoaie Teodor-Anton	Şc. cu cls. I-VIII Nr.56 "Jose Marti"	Bucureşti
10	Văcăroiu Andrei	C. N. "Mircea Cel Bătrân"	Vâlcea
11	Plop Teodor-Daniel	C. N. de Informatică "T. Vianu"	Bucureşti
12	Vlad Radu-Cristian	C. N. de Informatică "T. Vianu"	Bucureşti
13	Naiba Lucian	C. N. de Informatică "T. Vianu"	Bucureşti

Comisia este formată din:

Nr.	Nume şi prenume	Instituţia	Judeţ
1.	Emanuela Cerchez	Liceul de Informatică "Grigore Moisil"	Iaşi
2.	Constantin Gălăţan	C. N. "Liviu Rebreanu"	Bistriţa-Năsăud
3.	Alin Burţa	C. N. "B. P. Haşdeu	Buzău
4.	Mugurel-Ionuţ Andreica	Universitatea Politehnica	Bucureşti
5.	Dana Lica	C. N. "I. L. Caragiale" Ploieşti	Prahova
6.	Zoltan Szabo	Gr. Şc. "P. Maior" Reghin	Mureş
7.	Andrei Grigorean	Universitatea Bucureşti	Bucureşti
8.	Doru Popescu Anastasiu	C. N. "R. Greceanu" Slatina	Olt
9.	Marinel Şerban	Liceul de Informatică "Grigore Moisil"	Iaşi
10.	Moţ Nistor	C. N. "N. Bălcescu"	Brăila
11.	Adrian Diaconu	Universitatea Bucureşti	Bucureşti
12.	Mircea Paşoi	Universitatea Bucureşti	Bucureşti
13.	Tiberiu-Lucian Florea	Universitatea Bucureşti	Bucureşti
14.	Adrian Airinei	Universitatea "Al. I. Cuza"	Iaşi
15.	Daniel Păsăilă	Universitatea "Al. I. Cuza"	Iaşi
16.	Emilian Miron	Universitatea Bucureşti	Bucureşti
17.	Adrian Vladu	Brown University	SUA

prof. Emanuela Cerchez



Concursul Național de Informatică "Urmaşii lui Moisil"

http://www.liis.ro/~moisil2008

A devenit o tradiție ca înaintea Olimpiadei Naționale de Informatică Liceul de Informatică "Grigore Moisil" Iaşi să organizeze o "avanpremieră". Aflat la cea de a VIII-a ediție (respectiv cea de a doua în calitate de concurs național), "*Urmaşii lui Moisil*" îşi propune să reprezinte un instrument valoros de antrenare şi promovare a valorilor.

Anul acesta au participat în concurs 81 de elevi din 25 de județe şi 6 elevi din Republica Moldova. Concursul s-a desfăşurat pe site-ul de pregătire de performanță în informatică .campion, în condiții similare competițiilor internaționale de informatică.

.campion, un program de pregătire de performanță în informatică dezvoltat de Centrul Virtual de Excelență Siveco în colaborare cu Ministerul Educației şi Cercetării, este inițiat şi coordonat de profesorii Emanuela Cerchez şi Marinel Şerban de la Liceul de Informatică "Grigore Moisil" Iaşi. Ca element de noutate, în anul 2007-2008 .campion a oferit utilizatorilor un sistem de evaluare online, sistem de care bineînțeles au beneficiat şi participanții la Concursul Național "*Urmaşii lui Moisil*", ediția 2008.

Problemele de concurs – clasa a IX-a

hd

După cum ştiţi, spaţiul de memorare pe un hard-disk este partiţionat în discuri logice, fiecare partiţie (disc logic) având o anumită dimensiune. La rândul său, o partiţie este împărţită în mai multe zone de dimensiuni egale denumite clustere. Să considerăm că discul nostru are `N` clustere, numerotate de la `1` la `N`.

Un fişier memorat pe disc poate ocupa unul sau mai multe clustere, nu neapărat consecutive. Clusterele care nu sunt alocate nici unui fişier se numesc libere.

Pentru a mări viteza de acces la informaţiile de pe disc este de dorit ca toate clusterele care constituie un fişier să fie consecutive. Din acest motiv, este recomandabil ca o dată la 3-4 luni să realizaţi defragmentarea discului. Prin defragmentare, clusterele de pe disc sunt mutate, astfel încât la sfârşitul operaţiei de defragmentare, fiecare fişier va ocupa o secvenţă de clustere consecutive, alocate în ordine (adică primul cluster din secvenţă este primul cluster al fişierului, al doilea cluster din secvenţă este al doilea cluster al fişierului, etc.). Mai exact, fişierul `1` va avea alocate clusterele `1`, `2`, ..., p_1; fişierul `2` va avea alocate clusterele p_1+1, p_1+2, ..., p_1+p_2, etc., unde cu p_i am notat numărul de clustere alocate fişierului `i`.

Utilizatorii de Windows realizează defragmentarea cu un program special denumit *Disk Defragmenter*, dar voi nu aveţi acest program, aşa că va trebui să îl faceţi singuri.

Programul vostru poate să execute operaţii de mutare de clustere astfel:

- citeşte în memoria internă conţinutul unui cluster alocat unui fişier;
- scrie conţinutul clusterului citit într-un alt cluster liber.

După o operaţie de mutare, clusterul al cărui conţinut a fost citit în memorie şi apoi scris în celălalt cluster este declarat liber, iar clusterul în care am scris, evident, devine ocupat (fiind alocat fişierului).

Evident că programul vostru de defragmentare este considerat eficient dacă el va realiza defragmentarea utilizând un număr minim de operaţii de mutare.

Cerință

Să se determine numărul minim de operații de mutare necesare pentru defragmentarea unui disc.

Date de intrare

Fişierul de intrare `hd.in` conține pe prima linie un număr natural `N` reprezentând numărul de clustere de pe disc. Pe cea de a doua linie este scris un număr natural `F` reprezentând numărul de fişiere memorate pe disc. Pe următoarele `F` linii sunt descrise fişierele, câte un fişier pe o linie. Linia care descrie un fişier începe cu un număr natural `p` reprezentând numărul de clustere alocate fişierului. Urmează o succesiune de `p` numere naturale distincte cuprinse între `1` şi `N`, reprezentând clusterele alocate fişierului, în ordinea în care acestea au fost alocate. Numerele de pe aceeaşi linie sunt separate prin spaşiu.

Date de ieşire

Fişierul de ieşire `hd.out` va conține o singură linie pe care va fi scris un singur număr natural, reprezentând numărul minim de operatii de mutare necesare pentru defragmentarea întregului disc.

Restricții

$0 \leq F < N \leq 10000$

Toate clusterele alocate fişierelor sunt distincte.

Există cel puțin un cluster liber.

Exemplu

hd.in	hd.out	**Explicatie**
50 3 4 18 4 7 9 1 20 3 2 3 6	9	Fişierul 1 este format (în ordine) din clusterele 18, 4, 7 şi 9 care vor trebui mutate în clusterele 1, 2, 3, respectiv 4. Fişierul 2 este format dintr-un singur cluster (20) şi acesta trebuie mutat în clusterul 5. Fişierul 3 are clusterele 2, 3 şi 6 care trebuie mutate în clusterele 6, 7, 8. Numărul minim de mutări necesare este 9. O solutie posibilã cu 9 mutări este: 6->8, 2->6, 4->2, 9->4, 18->1, 20->5, 3->9, 7->3, 9->7

Timp maxim de execuție/test: 0.1 secunde

prof. Emanuela Cerchez, Liceul de Informatică "Grigore Moisil" Iaşi

cap-pajura

Monedele vechi aveau imprimate pe una dintre fețe figura regelui ("cap"), iar pe cealaltă un însemn al țării care a emis moneda ("pajura").
Atunci când arbitrul cere căpitanilor de echipă să aleagă jumătatea de teren pe care vor începe jocul, el aruncă în aer o monedă. Aceasta cade pe pământ cu una dintre fețe în sus, "capul" sau "pajura". Căpitanul de echipă care a ales fața de sus este cel care alege terenul.
Să presupunem că, în urma aruncării în sus a mai multor monede, acestea cad pe pământ într-o configurație de `nxm` monede (adică sub forma unei matrice cu `n` linii şi `m` coloane), unele având "capul" pe fața de sus, altele având "pajura". Arbitrul nostru, iubitor de probleme, dorește să obțină toate monedele cu pajura în sus. Pentru aceasta el restricționează operațiile posibile doar la două:
- o întreagă linie este "inversată";
- o întreagă coloană este "inversată".

Prin "inversarea" unei linii/coloane fiecare monedă de pe linia/coloana respectivă este inversată.

Cerință

Date fiind dimensiunile configurației `n` şi `m`, precum şi configurația monedelor, să se determine dacă, folosind numai cele două operații permise, se poate transforma configurația inițială astfel încât, la final, toate monedele să fie cu "pajura" în sus.

Date de intrare

Fişierul de intrare `pajura.in` conține mai multe seturi de date. Un set de date conține pe prima linie valorile `n` şi `m` separate printr-un spațiu. Următoarele `n` linii din setul de date conțin câte `m` caractere `C` sau `P`, fără spații între ele, reprezentând configurația inițială a monedelor. Seturile de date se termină cu două valori `0` separate printr-un spațiu.

Date de ieşire

Dacă în fişierul de intrare au fost `k` seturi de date, atunci fişierul de ieşire `pajura.out` va avea `k` linii, pe fiecare linie fiind scrisă una dintre valorile `1` sau `0`, după cum configurația inițială poate fi transformată sau nu.

Restricții şi precizări

$1 \leq n, m \leq 1000$

Numărul de seturi de date din orice fişier de intrare este ≤ 10.

Exemplu

pajura.in	pajura.out	*Explicații*
3 4 CPCC PCPP CPCC 5 2 PC CC CP CC CC 0 0	1 0	Fişierul de intrare conține 2 seturi de date. Pentru primul set de date o posibilă succesiune de operații este: CPCC coloana 2 CCCC linia 1 PPPP linia 3 PPPP PCPP ---------> PPPP ----------> PPPP ----------> PPPP CPCC CCCC CCCC PPPP Pentru cel de-al doilea set de date, nici o succesiune de operații nu conduce la un rezultat favorabil.

Timp maxim de execuție/test: 0.2 secunde

prof. Marinel Şerban, Liceul de Informatică "Grigore Moisil" Iaşi

Problemele de concurs – clasa a X-a

pc

În jocul cunoscut sub denumirea "Prețul corect" concurenții încearcă să ghicească prețul unui obiect. Câştigător este acel concurent care specifică un prEvenimente este prețul obiectului şi care este cel mai apropiat de prețul corect (pentru care diferența dintre prețul obiectului şi prețul specificat de concurent este minimă).

În această problemă prezentăm o versiune a acestui joc pentru un singur jucător.

Jucătorul are dreptul la G încercări pentru a ghici prețul corect şi V vieți.
După fiecare încercare, jucătorului i se spune dacă prețul a fost corect, prea mare sau prea mic.
Dacă la această încercare prețul a fost corect, jucătorul câştigă.
Dacă prețul nu a fost corect, jucătorul a consumat o încercare.
În plus, dacă prețul a fost prea mare, jucătorul a pierdut şi o viață.
Jucătorul pierde dacă a epuizat toate cele G încercări sau dacă la o încercare a spus un preț prea mare, dar a epuizat cele V vieți.
Prețurile sunt numere naturale nenule.
Pentru o pereche de valori G, V dată se poate determina o strategie astfel încât jucătorul să poată ghici sigur prețul dacă acesta este în intervalul [1,N].

Cerință

Scrieți un program care să determine pentru o pereche dată de valori G, V valoarea maximă a lui N astfel încât jucătorul să aibă o strategie sigură de câştig pentru orice preț din intervalul [1,N].

Date de intrare

Fişierul de intrare pc.in conține pe prima linie două numere naturale G şi V separate prin spațiu, cu semnificația din enunț.

Date de ieşire

Fişierul de ieşire pc.out va conține o singură linie pe care va fi afişat un număr natural reprezentând valoarea maximă determinată.

Restricții

1 ≤G ≤40
0 ≤V ≤40

Exemplu

pc.in	pc.out	pc.in	pc.out	pc.in	pc.out
3 0	3	3 1	6	7 7	127

Timp maxim de execuție/test: 0.1 secunde Evenimente

prof. Emanuela Cerchez, Liceul de Informatică "Grigore Moisil" Iaşi

sir

Fie numărul natural N şi şirul 1,1,2,2,3,3,...,N,N (şir crescător de lungime 2N, în care fiecare număr între 1 şi N apare de exact două ori). Cu numerele acestui şir construim două noi şiruri de lungime N. Fiecare număr din şirul iniţial se va depune fie în primul şir, fie în al doilea, astfel încât cele două şiruri rezultate să fie ordonate crescător.
De exemplu, pentru N=4 şi sirul iniţial 1,1,2,2,3,3,4,4, putem construi de exemplu şirurile 1,1,3,4 şi 2,2,3,4 sau şirurile 1,2,3,3 şi 1,2,4,4.

Cerinţă
Să se determine numărul de posibilităţi de împărţire a şirului iniţial în două şiruri de lungimi egale cu N, astfel încât ambele şiruri să fie crescătoare (nu neapărat strict), iar primul şir să fie întotdeauna mai mic sau egal din punct de vedere lexicografic decât al doilea. Pentru ca acest număr poate fi mare, se va determina numărul modulo 7001.

Date de intrare
Fişierul sir.in conţine pe prima linie numărul natural N.

Date de ieşire
Fişierul sir.out va conţine o singură linie pe care va fi scris un singur număr natural, reprezentând numărul total de posibilităţi, modulo 7001.

Restricţii şi precizări
$2 \le N \le 80$
Date două şiruri de lungime N, $a=a_1,a_2,\ldots,a_N$ şi $b=b_1,b_2,\ldots,b_N$, spunem că şirul a este mai mic lexicografic decât b, dacă există $1 \le k \le N$ astfel încât $a_1=b_1, a_2=b_2, \ldots, a_{k-1}=b_{k-1}$ şi $a_k<b_k$. Două şiruri de lungim<Evenimente exicografic dacă ele coincid.
Evenimente

Exemplu

sir.in	sir.out	**Explicaţii**

3	4	Şirul iniţial `1,1,2,2,3,3` poate fi împărţit în `4` moduri `(4 mod 7001 = 4)`:
		`1,1,2 2,3,3`
		`1,2,2 1,3,3`
		`1,2,3 1,2,3`
		`1,1,3 2,2,3`

Timp maxim de execuţie/test: 0.1 secunde

prof. Dan Pracsiu, Gr. Şc. "Ştefan Procopiu" Vaslui

Problemele de concurs – clasele a XI-a şi a XII-a

cadere

Drumurile dintre oraşele din Imperiul Roman stau la baza dezvoltării rapide a imperiului. De acest lucru şi-au dat seama şi inamicii imperiului care au început să le distrugă.

Înainte de a se lua măsuri se doreşte monitorizarea pagubelor provocate de aceste distrugeri. Unul dintre cei mai importanţi factori este distanţa faţă de capitală. Aceasta este numărul minim de drumuri ce trebuie parcurse pentru a ajunge în/din capitală (drumurile sunt bidirecţionale, iar lungimea lor nu contează).

Cerinţă

Date fiind drumurile dintre oraşe şi ordinea în care acestea sunt distruse, evaluaţi numărul de oraşe care s-au depărtat cu cel puţin o unitate după distrugerea fiecărui drum. Oraşele care nu mai au nici o legatură cu capitala se consideră la distanţa infinit.

Date de intrare

Fişierul de intrare `cadere.in` conţine pe prima linie patru numere naturale nenule `N`, `M`, `C` şi `Q`, reprezentând numărul deEvenimente ıl de drumuri iniţiale, capitala şi respectiv numărul de drumuri ce urmează a fi distruse.

Pe următoarele `M` linii se află câte două numere naturale: `O1` şi `O2`, reprezentând un drum între oraşele `O1` şi `O2`. Oraşele sunt numerotate de la `1` la `N`.

Pe următoarele Q linii se află, în ordine, drumurile ce urmează să fie distruse. Pe fiecare dintre cele Q linii sunt scrise două numere naturale O1 O2, reprezentând oraşele care sunt capetele unui drum (drum aflat în mulțimea drumurilor de mai sus). Numerele de pe aceeaşi linie sunt separate prin spații.

Date de ieşire

Fişierul de ieşire cadere.out va contine Q linii. Pe linia i va fi scris numărul de oraşe ce s-au mai depărtat de capitală cu cel puțin o unitate după distrugerea celui de-al i-lea drum din fişierul de intrare.

Restricții şi precizări

1 ≤ N ≤1 000
1 ≤ M ≤ 50 000
1 ≤ Q ≤ M

Între oricare două oraşe există cel mult un drum.
Un drum odată distrus nu se mai poate distruge din nou.

Exemplu

cadere.in	cadere.out	**Explicații**
5 6 2 2 1 2 1 4 2 5 2 3 2 4 4 5 1 2 2 4	1 2	După distrugerea drumului (1,2) oraşul 1 se departează cu o unitate de capitală (oraşul 2). După distrugerea drumului (2,4) oraşele 1 şi 4 se depărteaza cu câte o unitate.

Timp maxim de execuție/test: 1.5 secunde

Evenimente

ing. Marius Andrei, Google Inc

pioni

Nargy şi Fumeanu joacă un joc pe un graf orientat fără circuite cu N noduri (numerotate de la 1 la N) şi M arce. În acest graf există K pioni plasați în noduri. Fiecare jucător trebuie să mute, când îi vine randul, toți

pionii care se pot muta din nodurile în care se află în noduri adiacente. Mai exact, un pion situat în nodul `x` poate fi mutat în nodul `y`, dacă există arc de la `x` la `y`.
Cel care nu mai poate muta nici un pion atunci când îi vine rândul pierde partida. La începutul unui joc, prima mutare o face jucătorul Nargy.

Cerință

Dându-se mai multe configurații de pioni, să se determine cine câștigă jocul, dacă ambii jucători joacă optim.

Date de intrare

Fişierul de intrare `pioni.in` conține pe prima linie trei numere naturale `T N M`. Următoarele `M` linii vor conține câte două numere naturale `a b` cu semnificația că există arc de la `a` la `b`. Următoarele `T` linii vor descrie configurații de pioni astfel: primul număr de pe linie va fi `K` şi va fi urmat de `K` numere naturale reprezentând nodurile în care se afla pioni. Numerele de pe aceeaşi linie sunt separate prin spații.

Date de ieşire

Fişierul de ieşire `pioni.out` va conține `T` linii, câte o linie pentru fiecare configurație de pioni din fişierul de intrare. Pe linia `i` se va scrie numele jucătorului care câştigă (Nargy sau Fumeanu) pentru a `i`-a configurație de pioni din fişierul de intrare.

Restricții

`1 ≤T ≤5`
`1 ≤N ≤20 000`
`1 ≤M ≤50 000`
`1 ≤K ≤30 000`

Pot exista mai mulți pioni în acelaşi nod.

Evenimente

Exemple

pioni.in	pioni.out	**Explicație**
2 4 3 2 1	Nargy Fumeanu	1. Conform regulilor jocului Nargy trebuie să mute cu toți cei 3 pioni, iar singura posibilitate este să-i

3 1 4 3 3 2 2 3 2 1 4		mute pe toți în 1. Fumeanu nu mai poate muta. 2. Nargy muta pionul din 4 în 3, iar Fumeanu mută din 3 în 1. Acestea sunt singurele mutări posibile.

Timp maxim de execuție/test: 0.2 secunde

stud. Mircea Paşoi,
Universitatea Bucureşti,Facultatea de Matematică şi Informatică

Rezultate

Clasa A IX-A

Nr	Județ	Nume elev	Punctaj	Premiu
1	IS	Vicol Sergiu	200	Trofeul "Urmaşul lui Moisil" Premiul I
2	DJ	Cazacu Alexandru	190	Premiul II
3	PH	Tamaş Iulia	190	Premiul II
4	VN	Nicodei-Groper Eduard	160	Menţiune
5	IS	Albu Alexandru	100	Menţiune
6	TM	Sarca Adela	60	Menţiune

Clasa A X-A

Nr	Județ	Nume elev	Punctaj	Premiu
1	TM	Avramescu Andrei	30	Menţiune
2	NT	Mihăilă Ştefan	30	Menţiune
3	BZ	Sârbu Alexandru	30	Menţiune
4	Evenimente	Mihai	30	Menţiune
5	AG	Bivol Mihai	10	Menţiune

Clasa A XI-A

Nr	Județ	Nume elev	Punctaj	Premiu
1	IS	Paicu Alexandru	70	Premiul I
2	IS	Petrov Robert	10	Menţiune
3	BN	Retegan Alexandra	10	Menţiune

4	DB	Stancu Florin	10	Menţiune
5	VS	Zaiţ Victor	10	Menţiune

Clasa A XII-A

Nr	Judeţ	Nume elev	Punctaj	Premiu
1	MD	Berzan Constantin	130	Premiul I
2	AR	Schneider Ştefan	130	Premiul I
3	BC	Horlescu Marina	100	Premiul II
4	SV	Boca Lucian	90	Menţiune
5	NT	Munteanu Alexandru	80	Menţiune
6	IS	Tudose Vlad	70	Menţiune

prof. Emanuela Cerchez

Rezultatele elevilor de la Liceul de Informatică “Grigore Moisil” Iaşi la Olimpiada Naţională de Informatică 2008

Olimpiada Națională de Informatică pentru gimnaziu, Bacău, 5-8 februarie 2008:

Nr	Nume şi prenume	Clasa	Rezultat
1	Andrici Cezar	**5B**	**Menţiune**
2	Netedu Andrei	**5B**	**Menţiune**
3	Ţucăr Liana	**6A**	**Menţiune**
4	Rusu Alexandru	**6A**	**Menţiune**
5	Miron Alexandru	**5B**	
6	Filimon Marta	**6B**	
7	Ignat Andrei	**6B**	
8	**Blejuşcă Sabin**	6B	

De la stânga la dreapta: Liana Ţucăr, Andrici Cezar, Netedu Andrei şi Rusu Alexandru

Olimpiada Națională de Informatică pentru liceu, Ploieşti, 29 aprilie – 5 mai 2008:

Nr	Nume şi prenume	Clasa	Rezultat
1	Tudose Vlad	**12C**	**Premiul III** **Medalie de aur** **Calificat în lotul național**
2	Paicu Alexandru	**11D**	**Mențiune** **Medalie de argint**
3	Petrov Robert	**11B**	**Mențiune** **Medalie de argint**
4	Stoian Vlad	**11B**	**Mențiune** **Medalie de bronz**
5	Manea Vlad	**12C**	**Medalie de bronz**
6	Albu Alexandru	**9C**	**Medalie bronz**
7	Dominte Alexandru	**11C**	
8	**Chelariu Alexandru**	10D	

De la stânga la dreapta: Stoian Vlad, Manea Vlad, Albu Alexandru, Petrov Robert, Paicu Alexandru şi Tudose Vlad

prof. Emanuela Cerchez

INFOMATRIX 2008

"INFOMATRIX" este un concurs internațional de proiecte informatice organizat de Lumina institutii de invatamant, Universitatea din Bucureşti, Ministerul Educației Naționale, Cercetării şi Tineretului şi ISMB.

Liceul nostru a participat la acest concurs cu cinci echipaje, două echipaje la secțiunea **"Programming"** şi trei echipaje la secțiunea **"Digital Content".**

La secțiunea **"Programming"** elevii noştri au participat cu următoarele proiecte:

1. "HighS Jobs" – Chelariu Alexandru şi Pricopie Cosmin, din clasa a X-a D, coordonați de prof. Claudia Cărăuşu, proiect care a obținut medalia de bronz
2. "RStudent" – Niță Alexandru, din clasa a XI-a D, coordonat de prof. Dorina Ursache

La secțiunea **"Digital Content"**, proiectele participante au fost:

1. "Green Times" – Niță Alexandru şi Ignatov Iulia, din clasa a XI-a D, coordonați de prof. Dorina Ursache, proiect care a obținut medalia de argint

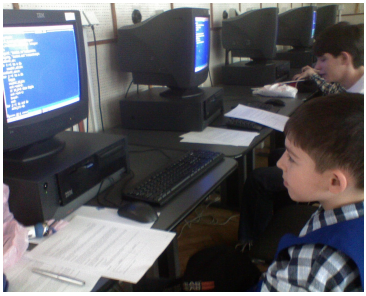
2. “The Anthropocene Epoch” – Hulubaş Radu şi Tabun Alexandra Ioana, din clasa a XI-a D, coordonați de prof. Silvia Grecu, obținând mențiune

3. “Impossible is nothing for them!” – Juncu Cătălin, Berigoi Daniel (clasa a XI-a E) şi Petrea Crina (clasa a XI-a D), coordonați de prof. Dorina Ursache, obținând medalia de bronz

Felicitări echipajelor pentru rezultatele obținute!

prof. Silvia Grecu

Concursul de Programare „Moisil++”



Concursul de programare „**Moisil++**”**,** desfăşurat la Liceul de Informatică „Grigore C. Moisil” în 26 mai 2008, s-a adresat elevilor de gimnaziu ai liceului şi a avut drept scop dezvoltarea spiritul competitiv, antrenarea concurenților în vederea participării cu succes la alte concursuri de informatică.

Concursul a avut trei nivele de dificultate, respectiv clasa a V-a, clasa a VI-a şi clasele VII-VIII. Programa de concurs a coincis cu aceea a fazei județeane a olimpiadei de informatică. La concurs au participat 31 de elevi din clasele de gimnaziu.

Din Comisia de organizare au facut parte elevii Robert Petrov, Alexandru Paicu si Vlad Stoian precum şi profesorii Cornelia Ivaşc, Oana Butnăraşu şi Ionel Maftei. Elaborarea subiectelor si evaluarea surselor concurenților au fost asigurate de elevii Robert Petrov, Alexandru Paicu şi Vlad Stoian, sub îndrumarea profesoarei Cornelia Ivaşc.

Rezultate 2008

Nume si prenume	Clasa	Premiul
Netedu Andrei	5B	**I**
Filimon Marta	6B	**I**
Rusu Alexandru	7A	**I**
Negruş Ştefan	8A	**I**
Pascaru Cosmin	5A	**II**
Andrici Cezar	5B	**II**
Ţucăr Liana	6A	**II**
Miron Alex	5B	**III**
Blejuşcă Sabin	6B	**III**

prof. Cornelia Ivaşc

Concursul “Al 4-lea val”

În luna mai s-a desfăşurat la Şcoala “Petru Poni” din Iaşi ediția a XI-a a concursului județean de informatică “Al 4-lea val”, organizat în colaborare cu fundația “Renaşterea Română” Iaşi.
Concursul a fost organizat pe două secțiuni:

- Secțiunea programare
- Secțiunea grafică/desen

Liceul nostru a participat cu două echipaje, corespunzător celor două secțiuni. Primul echipaj a fost format din elevele Țucăr Liana, clasa a VI-a A şi Filimon Marta, clasa a VI-a B, care au participat la secțiunea programare şi au obținut Premiul I.

Cel de al doilea echipaj a fost format din elevele Popa Ileana şi Vichilu Mădălina, din clasa a V-a A, care au participat la secțiunea grafică/design şi au obținut Mențiune.

Felicitări celor două echipaje pentru rezultatele obținute!

Însoțitor: prof. Simona Iuscinschi

Concursul de Informatică Aplicată

În perioada 23-25 mai 2008 am participat la etapa națională a Concursului de Informatică Aplicată, care s-a desfăşurat la Cluj. Acest cocurs a avut următoarele secțiuni:

- Secțiunea tehnologii, la care au participat:
  - elevi din clasele 9-10, indiferent de filieră, profil, specializare.
  - elevi din clasele 11-12 de la filiera tehnologică, profil tehnic, resurse şi protecția mediului, servicii.
- Secțiunea aplicații C#, la care au participat elevi din clasele 9-12 care au realizat proiecte în C#, indiferent de filieră, profilul sau specializarea la care studiază.

La Secțiunea Tehnologii concurenții au avut de parcurs trei probe.

Prima probă a constat dintr-un test grilă, cu un număr de 50 de întrebări, fiecare întrebare având patru variante de răspuns din care numai una singură corectă. În urma acesteia s-au calificat pentru susținerea probei a doua, primii 50% din elevii participanți la fiecare nivel de studiu. Timpul de lucru alocat a fost de 60 minute.

A doua probă, proba practică, a constat în rezolvarea pe calculator a subiectelor stabilite de comisia națională, timpul de lucru fiind de 45 minute. S-au calificat pentru proba a treia primii 50% din elevii participanți la proba a doua, la fiecare nivel de studiu.

Pentru cea de a treia probă, elevii au trebuit să prezinte un proiect multimedia, cu temă prestabilită. Timpul alocat prezentării unui proiect a

fost de 10 minute, iar printre criteriile de evaluare s-au numărat şi: originalitatea, realizarea artistică, conţinutul ştiinţific, uşurinţa în exploatare şi acurateţea tehnică.

Participarea la acest concurs a fost o experienţă interesantă pentru mine, având ocazia să particip la o confruntare naţională plină de fair play. Am fost impresionat de dotarea tehnică şi profesionalismul organizatorilor. În urma participării la acest concurs am obţinut menţiune, ceea ce consider că este un rezultat bun, având în vedere că este prima mea experienţă de acest gen.

Am avut multe de învăţat din acest concurs şi intenţionez să mă antrenez mai temeinic pentru concursurile din anii ce vor urma.

Tudorache Cristian
clasa a IX-a A

Stilul de programare

Stilul de programare (*coding style*) reprezintă un set de reguli pe care programatorii le respectă în scrierea programelor.

De ce este necesară definirea unui stil de programare?

Un secvență de cod care este bine scrisă va fi cu siguranță mai uşor de citit şi de înțeles atât pentru autorul ei, cât şi pentru membrii echipei cu care acesta lucrează. În consecință aplicația va fi mai uşor de întreținut, va conține mai puține erori şi de multe ori este mai compactă decât codul redactat "la repezeală", fără a respecta nici o regulă.

Formarea unui stil de programare este un obiectiv pe care un profesor de informatică trebuie să îl urmărească în mod constant, începând încă de la primele ore de algoritmi şi programare.

Reguli care definesc un bun stil de programare

Un bun stil de programare poate fi caracterizat prin trei cuvinte: simplu, clar, concis.

Identificatori

Identificatorii (nume) sunt utilizați pentru identificarea diferitelor elemente sintactice (variabile, constante, funcții, tipuri, etc).

Numele trebuie să fie semnificativ (cu alte cuvinte să ofere informații despre scopul cu care este utilizat elementul denumit), să fie scurt, uşor de memorat şi eventual să poată fi pronunțat.

Atunci când elementul denumit este global (deci poate fi accesat oriunde în cadrul aplicației) numele trebuie să descrie cât mai explicit posibil destinația acestuia (chiar este util să inserați un comentariu mai detaliat despre scopul elementului respectiv).

De exemplu:

```
int NrElevi; //numarul de elevi existenti in clasa
```

Când obiectul denumit este local (deci este utilizat doar în cadrul unei funcții), un nume scurt este suficient. Mai mult, în mod uzual sunt preferate denumirile `i`, `j`, k pentru indici într-un tablou; `p`, `q`, `r` pentru pointeri; `s`, `t` pentru şiruri de

caractere; `n`, `m` pentru dimensiunile unui tablou. Codul rămâne uşor de înţeles, dar devine mai concis.

Comparaţi următoarele două variante de însumare a elementelor vectorului `A`:

```
for (i=0; i<n; i++) Sum += A[i];
```

```
for (PozElementCurent = 0; PozElementCurent < NrTotalElemente; PozElementCurent++)
     Sum += A[PozElementCurent];
```

Nu trebuie să exagerăm cu utilizarea numelor lungi! De multe ori claritatea se poate obţine prin concizie!

Utilizarea majusculelor şi minusculelor în nume poate spori claritatea:

- numele constantelor sunt scrise numai cu majuscule;
- numele variabilelor formate prin alipirea mai multor cuvinte/prescurtări de cuvinte au iniţiala fiecărui cuvânt/fragment de cuvânt majusculă (stilul Pacal) – de exemplu: `Tablou`, `NrTelefon`; acest lucru nu este valabil şi pentru prima literă din numele varibilei (în stilul Java) – de exemplu: `tablou`, `nrTelefon`;
- pentru o portabilitate crescută, evitaţi ca numele variabilelor să înceapă cu _ (underscore), pentru a evita coincidenţele cu diferite variabile din bilbioteci;
- evitaţi utilizarea literelor ce pot fi confundate în contextul respectiv cu cifre (`0` şi `O`, `o` sau `1` şi `l`, `I`).

Construcţia numelor de variabile trebuie realizată în mod consistent.

De exemplu, atunci când utilizăm o coadă dacă am hotărât să o denumim `Coada`, atunci poziţia primului element din coadă va fi `IncCoada`, nu `IncC` sau `IncQ`. Mai mult, ori de câte ori utilizăm denumim elemente care au legătură între ele trebuie să includem în nume atât elementul comun, precum şi diferenţa dintre acestea.

De exemplu, voi denumi coada `Coada`, începutul cozii `IncCoada`, sfârşitul cozii `SfCoada`.

Numele funcţiilor trebuie să conţină un verb, care să indice acţiunea executată de funcţia respectivă. Dacă funcţia returnează un rezultat, este bine ca numele funcţiei să indice şi tipul rezultatului.

De exemplu, o funcție care testează dacă un număr este prim o vom denumi `EPrim()` (ceea ce indică şi acțiunea şi faptul că funcția va returna un rezultat logic).

O funcție care numără câte valori prime sunt într-un tablou o putem denumi `NumaraPrime()`.

Constantele semnificative din aplicația voastră ar trebuie să aibă nume semnificative. De exemplu, limitele maxime admise pentru tablouri:

```
#define NMAX 1000
```

```
const int NMAX 1000
```

Expresii

Pentru ca expresiile să fie uşor de citit şi de înțeles ar trebui să respectăm următoarele reguli:

1. Încadrați operatorii între spații. De exemplu: `x > 0 &&  x % 2 == 1`
2. Dacă ar putea exista neclarități referitoare la ordinea efectuării operațiilor, utilizați parantezele pentru a elimina neclaritățile. Chiar dacă parantezele nu sunt necesare, utilizați-le acolo unde considerați că ar aduce un plus de claritate. De exemplu: `x+1>>n` este mai bine să fie scrisă `(x+1)>>n` (operatorul + are prioritatea mai mare decât >>, dar din modul de scriere s-ar putea înțelege `x+(1>>n)`).
3. Nu parantezați în exces. De exemplu, `(((a)*(b)-(c))>0)` este o exagerare!
4. Divizați expresiile prea lungi în expresii mai scurte, o expresie prea lungă ar putea fi greu de înțeles.
5. Formulați expresiile în mod natural. De exemplu dacă vreți să verificați dacă o variabilă `x` are valoarea egală cu `1` nu scrieți `1 == x`, ci `x == 1`. Dacă vreți să verificați dacă indicele `i` este mai mic decât numărul de elemente din vector `n`, nu veți scrie `n > i`, ci `i < n` (cu alte cuvinte elementul variabil este bine să fie în membrul stâng al comparației, limita fixă fiind în dreapta).
6. Acordați a atenție deosebită operatorilor de incrementare şi decrementare. Utilizarea lor incorectă în cadrul expresiilor poate conduce la efecte secundare nedorite (deoarece modifică valoarea varibilei).

Instrucțiuni

Modalitatea de redactare e instrucțiunilor variază, în special în ce privește modul de scriere a acoladelor. Cea mai importantă regula rămâne: aplicați în mod consecvent aceleași reguli. Iată câteva dintre acestea:

1. Utilizați spațiile pentru o mai bună înțelegere! După numele instrucțiunii trebuie să urmeze spațiu. De asemenea după separatori (, ;) trebuie să urmeze spațiu.

Comparați aceste două variante:

```
for(i=0;i<n;i++)sum+=a[i];
```

```
for (i = 0; i < n; i++) sum += a[i];
```

2. Indentați! Atunci când o instrucțiune este subordonată unei alte instrucțiuni, deplasați instrucțiunea subordonată spre interior. De obicei acest lucru se realizează utilizând indentarea la un TAB (acționând o dată tasta TAB), ceea ce va asigura şi o aliniere pe verticală a instrucțiunilor.

Comparați cele două variante de scriere a instrucțiunii `if` din punctul de vedere al lizibilității:

```
if (a<b) if (a<c) if (c<b) x=1; else if (b>c) x=2; else x=3;
else x=4; else x=5;
```

```
if (a<b)
      if (a<c)
         if (c<b) x=1;
            else
            if (b>c) x=2;
                else x=3;
         else x=4;
       else x=5;
```

3. O instrucțiune compusă este încadrată între acolade. Există trei tehnici uzuale de împerechere a acoladelor. Important este ca tehnica pe care o preferați să fie utilizată în mod consistent.

O primă variantă este ca paranteza acoladă închisă `}` să fie plasată sub (pe aceeaşi coloană) cea deschisă `{`, ambele fiind indentate la un TAB (stilul GNU).

De exemplu:

```
for (i = 0; i < n; i++)
    {
```

```
     sum += a[i];
     p*=a[i];
    }
```

A doua variantă este ca acoladele să fie situate una sub alta, pe aceeaşi coloană, fără indentare (stil ANSI):

```
for (i = 0; i < n; i++)
{
     sum += a[i];
     p*=a[i];
}
```

A treia variantă este ca acolada deschisă să fie pe aceeaşi linie cu instrucţiunea care subordonează instrucţiunea compusă, iar acolada închisă să fie aliniată vertical cu numele instrucţiunii respective (stil Kernighan& Ritchie).

De exemplu:

```
for (i = 0; i < n; i++) {
     sum += a[i];
     p*=a[i];
}
```

4. O aceeaşi instrucţiune poate fi scrisă în numeroase moduri diferite. Utilizaţi modul cel mai "natural".

De exemplu:

```
for (i = 0; i < n; i++) sum += a[i];

for (i = 0; i < n; ) sum += a[i++];

for (i = n; --i > 0; ) sum += a[i];
```

Evident că prima variantă reprezintă modul cel mai clar şi natural de a parcurge un vector.

Comentarii

Inseraţi comentarii în program pentru o mai bună înţelegere a acestora.

Este indicat ca declaraţiile variabilele globale să fie însoţite de un comentariu referitor la scopul acestora. Funcţiile trebuie să conţină un comentariu referitor la acţiunile efectuate de funcţia respectivă. Şi, în general, orice bloc de program cu o funcţiune relativ independentă trebuie să fie însoţit de un comentariu.

Nu abuzaţi de comentarii! Nu comentaţi şi chestiuni evidente, aglomerând în acest mod codul şi făcându-l ilizibil!

Evident, codul trebuie să fie în concordanță cu comentariile asociate! În caz contrar cel puțin unul dintre acestea este greşit.

Bibliografie

1. Brian Kernighan, Rob Pike – "*Practice of programming*". Addison-Wesley, 1999.
2. Philips Medical Systems Software – "*Coding Standard C#*"
3. www.wikipedia.org – "*Coding style"*

prof. Emanuela Cerchez

Elevi şi profesori - creatori de soft educaţional

Elaborarea soft-ului educaţional este o provocare adresată atât elevilor, cât şi profesorilor. Deopotrivă creatori şi beneficiari, elevii şi profesorii, lucrând în echipă, redefinesc termenii învăţării.

Soft-ul educaţional nu a fost inventat astăzi. Preocupări în acest domeniu în comunitatea şcolară preuniversitară din România există încă din anii 80. De exemplu, în anul 1989 profesorul Marinel Şerban a brevetat două produse program denumite *Oscilaţii electromagnetice* şi *Mişcarea particulelor încărcate cu sarcină electrică în câmp magnetic*. Dar în anii 2000+ această preocupare tinde să devină un nou mod de abordare a creativităţii.

Pentru România există doi factori care au marcat decisiv evoluţia în acest domeniu:

- programul SEI, prin care noile tehnologii informaţionale au fost introduse în toate şcolile din România;
- Cupa Siveco, un concurs naţional de soft educaţional care are ca scop formarea creatorilor de *software* educaţional în scopul asigurării mentenanţei SEI.

An de an, peste 500 de proiecte participă la Cupa Siveco, dintre care 16-20 sunt selectate în Finala Cupei Siveco, unde se confruntă "pe viu", schimbă idei, dezvoltă colaborări viitoare.

De ce este pasionant să participi la Cupa Siveco?
Pentru că elevi şi profesori au ocazia de a lucra în echipă, *altfel*. Pentru că profesionalismul cu care elevii şi profesorii abordează crearea de soft educaţional îi aduce la nivelul marilor firme producătoare din domeniu (şi uneori peste acest nivel). Pentru că reprezintă o şansă de afirmare (deopotrivă pentru elevi şi profesori). Pentru că participând la Cupa Siveco devii membru al unei comunităţi, a creatorilor, a celor care vor să contribuie la societatea cunoaşterii.

Liceul de Informatică "Grigore Moisil" din Iaşi a avut şansa de a participa an de an, încă de la prima ediție, la Finala Cupei Siveco:

2003		
Chimie organică	Florin Chelaru, Radu Grosuleac, Corneliu Serediuc; prof. Magda Belța, prof. Cornelia Ivaşc	Proiectul a obținut premiul al II-lea la Cupa Siveco 2003
2004		

Heap interactiv	Bianca Milatinovici, Răzvan Grădinaru, prof. Emanuela Cerchez, prof. Marinel Şerban	Proiectul a obținut premiul al II-lea la Cupa Siveco 2004 şi a reprezentat România la ONLINE EDUGA Berlin 2004 [illegible] tehnologiilor educaționale.
2005		
Arbori parțiali de cost minim	Mihai-Alexandru Bîrsan, Andrei Diac, prof. Emanuela Cerchez, prof. Marinel Şerban	Proiectul a obținut premiul al II-lea la Cupa Siveco 2005. Echipa câştigătoare a participat la un *workshop* pe tema noi tehnologii educaționale la *Intel IT Innovation Center*, Dublin, Irlanda. De asemenea proiectul a fost prezentat în cadrul *Craig Barett World Tour*, martie 2006, Bucureşti, în fața reprezentanților guvernului, a mediului de afaceri şi mass-media din România.
Drum minim în graf	Andrei Chirilă, Cleonela Şerban, prof. Emanuela Cerchez, prof. Marinel Şerban	
Compuşi halogenați	Ştefan Macovei, Radu Nicolescu, Cezar Berea, prof. Magda Belța	
Independența de stat a României	Alexandru Veruzi, Ştefan Macovei, prof. Maria Rados	
2006		
Elemente de bază ale limbajului	Alexandru Periețanu, Dragoş Răducanu,	Câştigătorii Cupei Siveco 2006 Echipa a reprezentat România la

C/C++	prof. Emanuela Cerchez, prof. Marinel Şerban	ONLINE EDUCA Berlin 2006
Civilizații preistorice pe teritoriul României	Alexandru Veruzi, Alexandru Bucă, prof. Emanuela Cerchez	Proiectul a obținut premiul al II-lea la Cupa Siveco 2006
Matrice	Andrei Chirilă, Cleonela Şerban, prof. Gelu Susanu	
2007		
Conduita psihosocială	Alexandru Chimu, Vlad Constantinescu, prof. Claudia Cărăuşu, prof. Emanuela Cerchez	Proiectul a obținut premiul al II-lea la Cupa Siveco 2007, precum și Premiul de Popularitate la Conferința Națională de Învățământ Virtual 2007. Echipa câştigătoare a participat la un *workshop* pe tema noi tehnologii educaționale la *Intel IT Innovation Center*, Dublin, Irlanda.
Modelarea datelor	Alexandru Perieţanu, Alexandru Bucă, prof. Emanuela Cerchez, prof. Marinel Şerban	Proiectul a obținut premiul al III-lea la Cupa Siveco 2007
Procese psihice senzoriale Percepţii şi senzaţii	Cotelea Daniela, David George-Alexandru, prof. Adina Romanescu, prof. Cornelia Ivaşc	

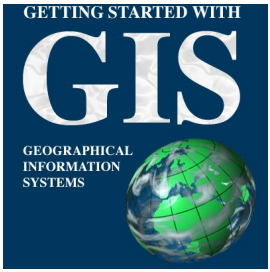
2008 Se vor califica la Cupa Siveco?	
Ora de dirigenție	Vlad Manea, Alexandru Perieţanu, Vlad Constantinescu prof. Claudia Cărăuşu, prof. Emanuela Cerchez
Permutări	Ionuț Fronea, George-Alexandru Ilaş, Silviu-Sergiu Hriscu prof. Lucian Lăduncă, prof. Emanuela Cerchez
Problema potrivirii şirurilor. Algoritmul KMP	Vlad Manea Vlad Tudose Andrei-Ionel Albeştean prof. Emanuela Cerchez, prof. Marinel Şerban

Le urăm SUCCES!

GIS – oportunitate educațională

prof. Emanuela Cerchez

În perioada 30 ianuarie – 4 februarie s-a desfăşurat la noi în liceu cursul intitulat "Introducere în ArcGis", organizat de ESRI România (Environmental Systems Research Institute, Inc.). La acest curs au participat elevi şi profesori de la Liceul de informatică "Grigore Moisil" din Iaşi şi de la Grupul Şcolar "Vasile Sav" din Roman.



GIS – Geographic Information System - este un sistem dedicat gestionării, analizei şi afişării informației geografice, ce este reprezentată de serii de seturi informaționale cum ar fi: hărți şi reprezentări globale, seturi de date geografice, modele de prelucrare şi de diagrame de lucru, modele de date şi metadate.

Printre componentele de bază ale unui sistem geografic informatic se numără persoanele, software-ul, datele, procedurile, hardwareul. Un sistem informatic geografic îndeplineşte funcții de introducere a datelor, stocare, interogare, analiză, afişare, rezultate.

ArcGIS reprezintă o colecție integrată de produse software GIS pentru construirea unui sistem GIS complet în cadrul unei organizații. Cadrul de lucru al arhitecturii ArcGIS permite distribuirea funcționalităților GIS pe orice platformă, fie desktop, servere (inclusiv Web), sau instrumentele mobile. Această arhitectură împreună cu bazele de date geospațiale (geodatabase) oferă instrumentele necesare implementării unui sistem informatic geografic inteligent.

ArcGIS este pus la dispozitie de către unicul distribuitor autorizat pentru teritoriul României al ***Environmental Systems Reseα***Din lumea informaticii liderul mondial în furnizarea aplicatiilor software GIS.

Cele mai utilizate produse ArcGIS din categoria desktop:

1. ArcView este o aplicație software GIS ce permite vizualizarea, cartografierea, crearea şi analiza datelor geografice. Folosind ArcView puteți înțelege contextul geografic al datelor dumneavoastră. Aceasta aplicație oferă totodată suportul decizional necesar pentru rezolvarea cat mai rapidă a problemelor şi luarea corectă a deciziilor.

ArcView este cea mai utilizată aplicație software GIS din clasa desktop deoarece oferă o modalitate foarte uşoară de utilizare a datelor geografice. Dispunând de un număr variat de simboluri şi de capacități cartografice, veți putea foarte uşor crea hărți de calitate ridicată. ArcView vă uşurează gestionarea şi editarea datelor în cadrul organizației. De fapt, orice producător de date geografice vă poate oferi datele într-un format compatibil cu ArcView.

2. ArcEditor reprezintă un sistem complet GIS desktop pentru editarea şi administrarea datelor geografice. ArcEditor face parte din familia de produse ArcGIS şi include în plus față de funcționalitatea ArcView, instrumente de editare GIS mult mai complexe.

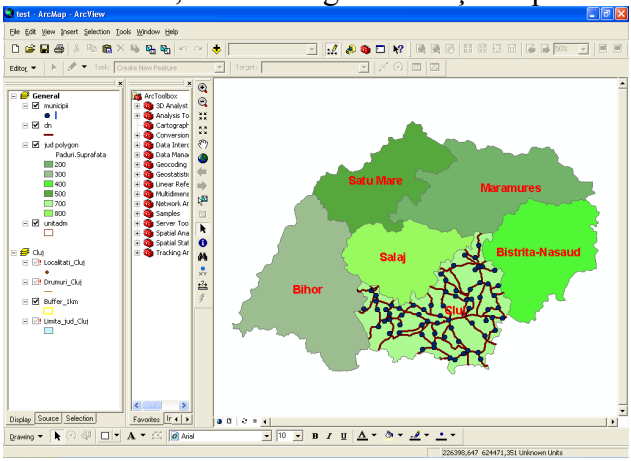
3. ArcInfo este cea mai completă soluție GIS disponibilă. Această aplicație include atât funcționalitatea ArcView cât şi pe cea din ArcEditor, şi în plus o funcționalitate avansată de conversie a datelor şi geoprocessing. Profesioniştii GIS utilizează ArcInfo pentru construirea datelor spațiale, modelarea şi analiza acestora şi afişarea hărților şi rapoartelor rezultate.

ArcInfo oferă întreaga funcţionalitate pentru crearea şi gestionarea unui GIS inteligent. Această funcţionalitate este accesibilă prin intermediul unei interfeţe foarte uşor de utilizat ce poate fi particularizată şi extinsă prin intermediul modelelor, script-urilor şi a aplicaţiilor.

ArcView pune la dispoziţia utilizatorilor diverse aplicaţii, precum:

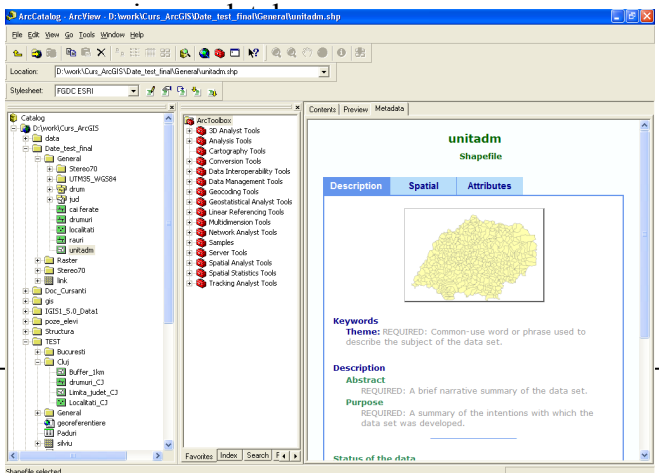
1. ArcMap, ce permite

- vizualizarea aplicaţiei prim:Din lumea informaticii
- realizarea hărţii, prin operaţii precum: edıtare, ınterogare, analiză, realizarea graficelor şi a rapoartelor



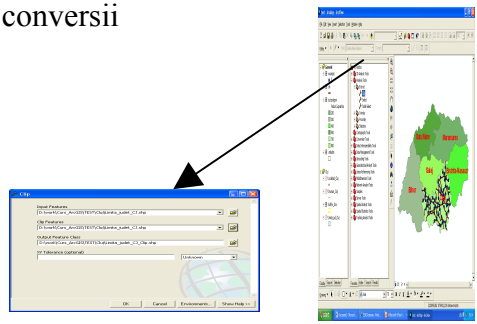
2. ArcCatalog

- organizarea şi gestionarea bazei de date
- crearea şi vizualizarea documentaţiei datelor (metadata)



3. ArcToolbox

- este o fereastră disponibilă în ArcCatalog şi ArcMap
- oferă funcţii de procesare geografică: managementul datelor, analize şi conversii



Promovarea tehnologiei GIS în şcoală se poate realiza prin activităţi de informare, susţinere de ore interdisciplinare (geografie-informatică), activităţi de colaborare cu alte şcoli sau instituţii care utilizează acestă tehnologie precum şi prin introducerea unor discipline opţionale pe acestă tematică.

Există multiple avantaje ale promovării tehnologiei GIS în şcoală. Printre acestea se numără iniţierea viitorilor utilizatori, formarea specialiştilor, dezvoltarea unor parteneriate viabile între şcoli şi instituţii prin implicarea elevilor în realizarea de proiecte GIS utile comunităţii, formarea abilităţilor de lucru în echipă, realizarea contactului cu sfera socio-profesională şi cu cerinţele actuale ale societăţii.

În liceul nostru s-au derulat două proiecte finanţate de Uniunea Europeană care au folosit această tehnologie:

- "Şanse egale în utilizarea tehnologiilor moderne", proiect multilateral Comenius, **www.liis.ro/~comenius**
- "GIS -tehnologie moderna pentru bune practici în şcoală", proiect de mobilităţi Leonardo da Vinci

Bibliografie:

- Introducere în ArcGis – suport de curs
- www.esriro.ro
- Wilpen L. Gorr, Kristen S. Din lumea informaticii :ond Edition

prof. Simona Iuscinschi

Macbook Air

În cursul acestui an a fost lansată o nouă jucărie de către firma Apple. Aceasta impresionează prin subţirime şi sistemul multi-touch.

Cele mai subţiri zone ale lui MacBook Air măsoară numai **0.16 inch**, în timp ce zonele în care este cel mai gros ajung la maxim 0.75 inch. Sistemul **multi-touch** nu este implementat în ecran aşa cum ne-am fi putut aştepta, ci **pe trackpad** . Cei care au avut ocazia de a-l testa spun că sistemul merge foarte bine, cu menţiunea că **nu poate fi folosit (deocamdată) în toate programele**.

MacBook Air este dotat cu procesoare **Intel Core 2 Duo** de 1.6 sau 1.8GHz şi **2 GB RAM** DDR2 la 667 MHz. Harddisk-ul are 80 GB, 4200

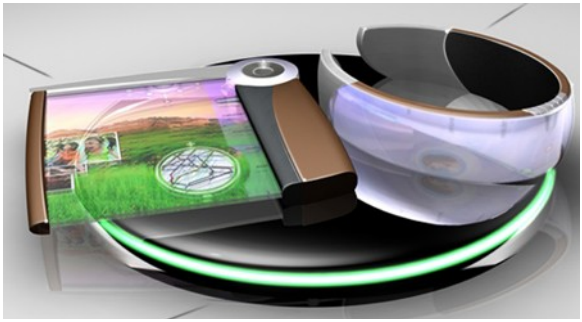
rpm şi este conectat pe PATA. Opţional, se poate achiziţiona un **SSD** ultimul răcnet, de 64 GB. Pe partea de conectivitate are suport Wi-Fi şi bluetooth, **un singur port USB** şi un Micro-DVI. Bateria ţine 5 ore, conform producătorului.

Ca minusuri, pe lângă faptul că are un singur port USB MacBook Air **nu are Ethernet**. **Componentele de bază** precum harddisk-ul, memoria RAM sau bateria **nu pot fi schimbate de utilizator**. Evident că Apple Din lumea informaticii ocui bateria pentru 129 de dolari.

Ursachi Răzvan, clasa a IX-a D

FLUX PC, calculatorul viitorului

Flux PC este un concept de calculator de generaţie următoare, care vă va lăsa cu siguranţă uimiţi. Calculatorul ultra-portabil este compus din 3 părţi - brăţara, ecranul pliabil şi incărcătorul - şi poate fi purtat pe încheietură. Conceptul urmează principiile flexibilităţii, mobilităţii, a interfţei prietenoase şi, nu în ultimul rând, a modei.



Brăţara stochează informaţia personală a utilizatorului şi se conectează wireless la ecranul pliabil de 25 x 30 x 1 cm, cu touchscreen. Calculatorul se poate închide şi deschide cu uşurinţă. Odată pliat, poate fi purtat în buzunar, în poşetă sau pe încheietură.

Mai mult decât atât, Flux PC poate lua, după gust, aspectul unei braţări de aur, ale şi modele preferate de uti tfel că dimneaţa, când mâna.

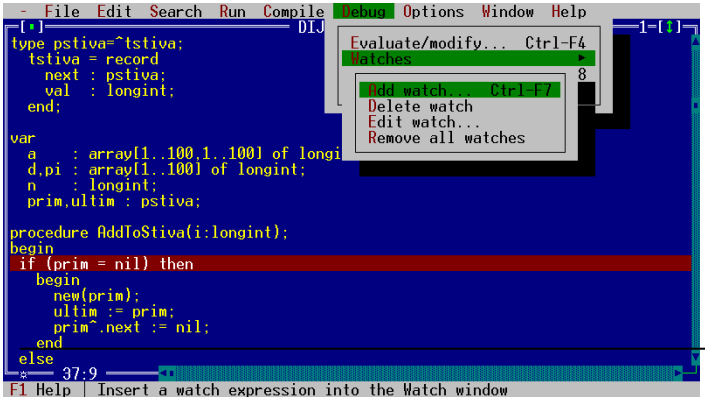
Melinte Mădălina, clasa a IX-a D

Borland Pascal 7.0

Dragul nostru Borland Pascal ştie multe să facă, păcat că nu merge CRT-ul pentru calculatoarele de după neoliticul superior. Isteț program, păcat că nu poate sa îți explice toate erorile pe care le dă, iar când se blochează (din vina lui, nu din vina utilizatorului!) de multe ori nu merge decât închis, CTRL+Break merge doar în cazurile fericite. Şi cui nu îi plac finalurile fericite? Pascalului, normal!

Unul din predecesorii lui Borland Pascal

ERROR 123.BB:" [or { not used"
ERROR 2112:"You have brown hair"
ERROR 231:"There is no error"



CRT

Error - division by 0. You can't write about this! Windows is shuting down... or you don't use it!

Fişiere

Din lumea informaticii pentru ajutorul adus, să nu mai fim nevoiți să scriem pe "fond negru". Numai să nu uităm să îl strigăm

(ReadLn(fin,n) sau WriteLn(fout,n), că altfel se supără şi apare iar fondul negru, ignorând toate străduințele noastre de a-i face loc printre variabile. La citire, trebuie să ştim cât şi ce avem de citit, că altfel, el nu se supără şi ia oricâte numere, dar scrise degeaba, deoarece programul işi ia DOAR numerele de care are nevoie.

If fişier NOT STRIGAT Then
Error 23:Fond Negru;

Char şi String

Două persoane care ne permit să mai scăpăm de cifre şi iar cifre, permițându-ne să scriem şi cu "a", "b", "c"... "<", "> ", "@ ". String e lung, de 255 de caractere, dar Char e util doar în anumite cazuri, de exemplu când trebuie să citim spații între numere.

Vectori şi matrici

Îi putem asemăna cu Stan şi Bran. Unui înalt şi altu-i lat. Pentru că matricea-i mai lată, nu încape, ca Stan vectorul, doar pe V:ARRAY[1..100] of Integer, ci îi trebuie M:ARRAY[1..100,1..100] of Integer. Din păcate, Pascalul nu are şi instrucțiuni de slăbire.

Râme

Am mai auzit de viermi electronici, dar nu de "rame de Pascal". Formate din ce caractere vrea "creatorul" se plimbă pe ecran până se lovesc de marginea de sticlo-număr al "For"-ului. Dar, din nou, trebuie ca mr. CRT să binevoiască să îşi facă apariția... Iar numărul de kilometri pe oră al râmei depinde de câți cai putere are Delay-ul. Cu cât Delay are o greutate mai mică, cu atât viteza creşte, adică invers proporțional.

Microsoft Word 2003

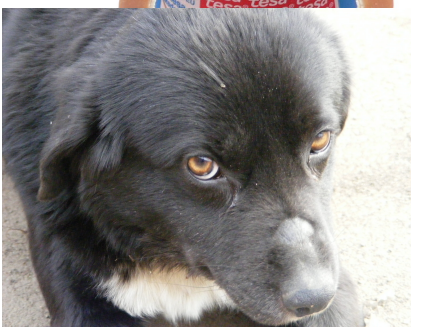
Un cuvânt care îl au pe buze majoritatea redactorilor de la ziare, sau cei care au nevoie să facă un document în format electronic, sau cei care doresc un document pe care clasica maşină de scris nu îl poate face aşa cum trebuie. Nu spun că nu există şi alte programe de scris, precum "carneţelul" Notepad, numai că MiDin lumea informaticii nplex, permiţând executarea mai multor operaţii, cum ar fi stabilirea unui Background. Desigur, programul a avansat pe parcursul timpului, comparând Microsoft Word 2000 cu Microsoft Word 2003, cel din urmă fiind mai complex decât predecesorul său, bătrânul 2000.

Un lucru pe care maşina de scris nu îl poate face, este să "lipească" poze.

Doar în cazul în care nu doriţi să folosiţi acest material modern ... scotch-ul.

Dacă doriţi sa puneţi puneţi poze în Word, uitaţi ce simplu este!

Vedeţi scotch pe lângă poză?
Normal că nu! Word-ul foloseşte scotch transparent!



Un lucru bun totuşi. Maşina de scris (în cazul în care nu aveţi hârtie colorată şi doriţi un fond) vă lasă să vă exprimaţi talentele artistice.

OBS.

Vă sfătuim să coloraţi planşa înainte de a o băga în maşina de scris. Mulţumim.

Trebuie conştientizat totuşi faptul că Microsoft Word este mai eficient decât maşina de scris, nu numai prin faptul că aveţi mai multe

moduri de făcut un document în Word, dar şi pentru că scap de bătaia de cap de a găsi diferenţe, avantaje şi dejavantaje.

Ce-am ajuns, de râsul unui morman de fire... Ce ţi-i şi lumea azi...”-Interviu cu o Maşină de Scris.

A se vedea de acelaşi autor articolul Probleme rezolvate lgii”.

Locovei Bogdan, clasa a VI-a A

Problemă rezolvată – Bile

Olimpiada Naţională de Informatică, Ploieşti 2008

Bogdan a primit de ziua sa un joc foarte ingenios. Jocul este constituit dintr-o cutie cu două compartimente. Iniţial în primul compartiment se află k bile (numerotate de la 1 la k), iar în al doilea compartiment se află n-k bile (numerotate de la k+1 la n).
Cele două compartimente comunică printr-o uşiţă basculantă specială care are două lăcaşuri. Un lăcaş se află în compatimentul 1, iar celălalt în compartimentul 2. Într-un lăcaş poate să încapă o singură bilă.
Vasile poate alege o bilă din compartimentul 1 şi o bilă din compartimentul 2, să plaseze bilele alese în cele două lăcaşuri ale uşiţei şi să rotească uşiţa. Astfel bila din compartimentul 1 va trece în compartimentul 2, iar bila din compartimentul 2 va trece în compartimentul 1.
Aceasta este singura mutare posibilă.
Scopul jocului este de a executa o succesiune de mutări astfel încât în compartimentul 1 să se obţină succesiv toate submulţimile distincte de k elemente ale mulţimii {1, 2, ..., n}.

Cerinţă

Scrieţi un program care să afişeze submulţimile de k elemente ale mulţimii {1, 2, ..., n} în ordinea în care acestea pot fi obţinute în compartimentul 1 cu ajutorul uşiţei basculante.

Date de intrare

Fişierul de intrare bile.in va conţine pe prima linie numerele naturale n şi k, separate printr-un spaţiu.

Date de ieşire
Fişierul de ieşire bile.out va conține câte o linie pentru fiecare submulțime obținută în compartimentul 1. Pe fiecare linie vor fi scrise în ordine crescătoare k numere naturale din mulțimea {1, 2, ..., n}, separate prin câte un spațiu, reprezentând elementele submulțimii. Pe prima linie va fi afişată submulțimea inițială (adică numerele 1 2 ... k).
Restricții şi precizări
$1 \leq k < n \leq 20$ Probleme rezolvate
Soluția nu este unică, puteți afişa oricare dintre variantele corecte.

Exemplu

bile.in	bile.out	*Explicație*
4 2	1 2 1 3 1 4 2 4 2 3 3 4	La prima mutare au fost plasate în uşița basculantă bila 2 (din compartimentul 1) şi bila 3 (din compartimentul 2). La a doua mutare au fost alese bilele 3 şi 4. La a treia mutare au fost alese bilele 1 şi 2. La a patra mutare au fost alese bilele 4 şi 3. Iar la ultima mutare au fost alese bilele 2 şi 4.

Timp maxim de execuție/test: 1.4 secunde (pentru Windows şi Linux)
Pentru Linux, total memorie disponibilă 5 MB, din care 1 MB pentru stivă.
Descrierea soluției
Soluție 1 (prof. Emanuela Cerchez)
Problema cere să generăm toate submulțimile de k elemente ale mulțimii {1, 2, ..., n} astfel încât oricare două submulțimi generate consecutiv să difere printr-un singur element.
Numărul de soluții este, evident, Comb(n,k)=n!/(k!*(n-k)!).
Algoritmul de generare este inspirat din relația de recurență a lui Pascal: Comb(n,k)=Comb(n-1,k)+Comb(n-1,k-1).
Mai exact, submulțimile de k elemente ale mulțimii {1, 2, ..., n} pot fi partiționate în două clase: submulțimile care conțin valoarea n (acestea sunt Comb(n-1,k-1)) şi submulțimile care nu conțin valoarea n (acestea sunt Comb(n-1,k)).

Să notăm cu Sn,k o soluție corectă (adică o succesiunea combinărilor de n luate câte k care respectă condițiile din enunț).
$S_{n,1}$={1} {2} {3}... {n}, pentru orice n
$S_{n,n}$={1, 2, 3, ..., n}
Pentru a genera $S_{n,k}$:

1. vom genera combinările de n-1 luate câte k în ordinea specificată (adică $S_{n-1,k}$)
2. vom genera combinările de n-1 luate câte k-1 în ordinea specificată (a[illegible] elementul n
3. Construim $S_{n,k}$ din $S_{n-1,k}$ urmat de $\underline{S_{n-1,k-1}}$U{n} unde cu $\underline{S_{n-1,k-1}}$ am notat $S_{n-1,k-1}$ considerat în ordine inversă.

De exemplu:
$S_{2,1}$={1}{2}
$S_{3,1}$={1}{2}{3}
$S_{3,2}=S_{2,2}$ $S_{2,1}$U{3}={1,2}{2,3}{1,3}
$S_{4,1}$={1}{2}{3}{4}
$S_{4,2}=S_{3,2}$ $S_{3,1}$U{4}={1,2}{2,3}{1,3}{3,4}{2,4}{1,4}
$S_{4,3}=S_{3,3}$ $S_{3,2}$U{4}={1,2,3} {1,3,4} {2,3,4}{1,2,4}
$S_{5,2}=S_{4,2}$ $S_{4,1}$U{5}={1,2}{2,3}{1,3}{3,4}{2,4}{1,4}{4,5}{3,5}{2,5}{1,5}
$S_{5,3}=S_{4,3}S_{4,2}$U{5}={1,2,3}{1,3,4}{2,3,4}{1,2,4}{1,4,5}{2,4,5}{3,4,5}{1,3,5}{2,3,5}{1,2,5}
etc
Se poate demonstra cu uşurință prin inducție că acest procedeu produce o secvență de combinări care respectă condițiile din enunț.
Observați că procedeul de generare este asemănător celui utilizat pentru codul *Gray*.
Vom descrie un algoritm de tip succesor pentru generarea $S_{n,k}$:
Configurația inițială este s={1,2,...,k}
Cât timp nu am generat toate combinările:

- afişăm configurația curentă;
- generăm configurația următoare, dacă există.

Generarea succesorului:
Pentru a nu trata în mod special cazul ultimului element, s[k+1]=n+1;
for (j=1; j<=k && s[j]==j; j++);
if (j%2!=k%2)

```
   if (j==1) s[1]--;
     else {s[j-1]=j; s[j-2]=j-1;}
   else
   if (s[j+1]!=s[j]+1) {s[j-1]=s[j]; s[j]++;}
         else {s[j+1]=s[j]; s[j]=j;
```

Implementare în limbajul C:

```
#include <stdio.h>
#define InFile "bile.in"
#define OutFile "bile.out"
#define NMax 50

int n, k;
int s[NMax];
long nr;

long Pascal(void);
int main()
{FILE * fout=fopen(OutFile,"w");
 FILE * fin=fopen(InFile,"r");
 long int i, j;
 fscanf(fin,"%d %d",&n,&k);
 fclose(fin);
 for (i=1; i<=k; i++) s[i]=i;
 s[k+1]=n+1;
 nr=Pascal();
 for (i=0; i<nr; i++)
     {
     //afisare
     for (j=1; j<k; j++) fprintf(fout,"%d ",s[j]);
     fprintf(fout,"%d\n",s[k]);
     //succesorul
     for (j=1; j<=k && s[j]==j; j++);
     if  (j%2!=k%2)
        if (j==1) s[1]--;
          else {s[j-1]=j; s[j-2]=j-1;}
```

```
        else
        if (s[j+1]!=s[j]+1) {s[j-1]=s[j]; s[j]++;}
            else {s[j+1]=s[j]; s[j]=j;}

    }
 fclose(fout);
 return 0;
}
```

Probleme rezolvate

```
long Pascal(void)
{
int i, j;
long L1[NMax], L2[NMax];
L1[0]=L1[1]=L2[0]=1;
for (i=2; i<=n; i++)
   {
    for (j=1; j<i; j++)
       L2[j]=L1[j-1]+L1[j];
    L2[i]=1;
    for (j=1; j<=i; j++)
       L1[j]=L2[j];
   }
return L2[k];
}
```

Soluție 2 (prof. Zoltan Szabo, Gr. Şc. „P. Maior" Reghin)

Să generăm combinările cu metoda *backtracking*, folosind o stivă st în care vom pune k elemente în ordine strict crescătoare, generând toate soluțiile în ordine lexicografică.

Putem observa, că în toate cazurile în care ultimul element al stivei are succesor (st[k]<n), cerința cerută va fi respectată (fiindcă ultimul elemente creşte cu 1, şi toate celelalte elemente rămân neschimbate).

În cazurile în care ultimul element al mulțimii nu are succesor (st[k]=n), trecerea către următorul element, nu va respecta cerința dorită.

Să observăm, că elementul st[p] nu are succesor în ordine lexicografică, dacă st[p]-p=n-k. Deci valorile posibile pentru st[p] sunt incluse în intervalul [st[p-1], n-p-k].
Vom modifica algoritmul anterior, astfel încât elementele stivei să poată să aibă la diferite nivele paşii 1 sau -1 astfel:

- pentru pasul 1 valoarea st[p]=n-p-k nu are succesor;
- pentru pasul -1 valoarea st[p]=st[p-1] nu are predecesor .

Probleme rezolvate

Combinări în ordine lexicografică	*Pasul corespunzător fiecărui element din stivă*	*Generarea bilelor*	*Pasul corespunzător fiecărui element din stivă*	*Explicație*

1 2 3 4 1 2 3 5 1 2 3 6 1 2 4 5 1 2 4 6 1 2 5 6 1 3 4 5 1 3 4 6 1 3 5 6 1 4 5 6 2 3 4 5 2 3 4 6 2 3 5 6 2 4 5 6 3 4 5 6	(1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1) (1,1,1,1)	1 2 3 4 1 2 3 5 1 2 3 6	(1,1,1,1) (1,1,1,1) (1,1,1,1)	Pornim cu prima combinare banală şi pt. fiecare nivel pas=1
		1 2 4 6 1 2 4 5	(1,1,1,-1) (1,1,1,-1)	Când ultimul nu mai are succesor, revenim, generăm succesorul, iar pentru elementele schimbăm pasul
		1 2 5 6	(1,1,1,1)	Penultimul şi ultimul nivel nu mai au succesor
		1 3 5 6 1 3 4 6 1 3 4 5 1 4 5 6 2 4 5 6 2 3 5 6 2 3 4 6 2 3 4 5 3 4 5 6	(1,1,-1,-1) (1,1,-1,-1) (1,1,-1,-1) (1,1,1,1) (1,-1,-1,-1) (1,-1,-1,-1) (1,-1,-1,-1) (1,-1,-1,-1) (1,1,1,1)	La nivelul 2 am trecut la succesor, iar nivelele superioare şi-au schimbat pasul Etc

Probleme rezolvate

Să observăm că prın acest procedeu, după ce am generat un element, restul elementelor stivei se pot completa instantaneu până la nivelul k. Astfel avem un algoritm *backtracking* optimizat, care generează toate soluțiile în ordine corectă fără să treacă prin valori succesive nevalide.

Implementare în limbajul Pascal

```
program bactracking;
const fin='bile.in';
      fout='bile.out';
```

```
type stiva=array[1..23] of integer;
var st,pas:stiva;
    k,n,i:integer;
    as,este:boolean;
    f:text;
procedure tipar;
var i:integer;
begin
  for i:=1 to k do write(f,st[i],' ');
  writeln(f);
end;

procedure revine;
var p:integer;
begin
p:=k;
while (p>0) and((pas[p]=1) and (st[p]-p=n-k)) or
      ((pas[p]=-1) and (st[p]=st[p-1]+1)) do dec(p);
if p=0 then este:=false
   else
   begin
   este:=true;
   if pas[p]=1 then
      begin
      inc(st[p]);
      while p<k do
           begin
           inc(p);                                    Probleme rezolvate
           if st[p]-p=n-k then pas[p]:=-1
              else begin pas[p]:=1; st[p]:=st[p-1]+1; end;
           end;
         end
        else
        begin
        dec(st[p]);
```

```
        while p<k do
          begin
          inc(p);
          if st[p]-p=n-k then pas[p]:=-1
            else begin pas[p]:=1; st[p]:=st[p-1]+1; end;
          end;
        end
      end;
end;
begin
  assign(f,fin);  reset(f);
  readln(f,n,k);
  close(f);
  assign(f,fout); rewrite(f);
  for i:=1 to k do begin st[i]:=i; pas[i]:=1; end;
  tipar;
  repeat
    revine;
    if este then tipar;
  until not este;
close(f);
end.
```

autor: prof. Emanuela Cerchez
Liceul de Informatică „Grigore Moisil” Iaşi

Problemă rezolvată – Borcane

Olimpiada Națională de Informatică, Ploieşti 2008
baraj lot național

autor: prof. Marinel Şerban
Liceul de Informatică „Grigore Moisil” Iaşi

Pe perioada vacanței, Bogdan s-a angajat vânzător la o cofetărie. Aici bomboanele sunt păstrate în `n` borcane, numerotate de la `1` la `n`. Din când în când, de plictiseală, Bogdan alege două borcane, ia câte o bomboană din fiecare borcan ales şi apoi pune cele două bomboane într-un al treilea borcan. În aşteptarea clienților, Bogdan studiază următoarea problemă: este posibil ca prin astfel de mutări să adune toate bomboanele într-un singur borcan?

Cerință

Dat fiind numărul de borcane şi numărul de bomboane din fiecare borcan, scrieți un program care să determine o succesiune de mutări de tipul celei descrise în enunț prin care toate bomboanele să fie adunate într-un singur borcan.

Date de intrare

Fişierul de intrare `borcane.in` conține pe prima linie numărul natural `n`, reprezentând numărul de borcane. Pe cea de a doua linie sunt scrise `n` numere naturale $b_1\ b_2\ \ldots\ b_n$, separate prin câte un spațiu, reprezentând, în ordine, numărul de bomboane din fiecare borcan.

Date de ieşire

Fişierul de ieşire `borcane.out` va conține în ordine mutările executate, câte o mutare pe o linie. O mutare este descrisă prin 3 numere naturale separate prin câte un spațiu `a b c` cu semnificația: "*se ia câte o bomboană din borcanele* **a** *şi* **b** *şi se plasează cele două bomboane în borcanul* **c**".

Restricții

- $4<=n<=100$
- $0<=b_i<=1000$
- $b_1+b_2+\ldots+b_n>=4$
- inițial există cel puțin două borcane care conțin bomboane

Probleme rezolvate

Exemplu

bomboane.in	bomboane.out	Observații
4 2 2 2 2	1 2 4 2 3 4	Inițial, sunt 4 borcane care conțin 8 bomboane. O posibilă soluție este:

	1 3 4	▪ se ia câte o bomboană din borcanele 1 şi 2 şi se pun în borcanul 4: 1 1 2 4 ▪ se ia câte o bomboană din borcanele 2 şi 3 şi se pun în borcanul 4: 1 0 1 6 ▪ se ia câte o bomboană din borcanele 1 şi 3 şi se pun în borcanul 4: 0 0 0 8 În final toate cele 8 bomboane se vor găsi în borcanul 4.

Timp maxim de execuţie: 0.1 secunde/test
Memorie totală disponibilă 2 MB , din care 1 MB pentru stivă.

Descrierea soluţiei
Soluţie1- *prof. Marinel Şerban*
În primul rând vom demonstra că problema are întotdeauna soluţie.
Să notăm cu
$m=b_1+b_2+...b_n$ numărul total de bomboane.
Vom demonstra prin inducţie după m propoziţia:
P(m) = ”pentru orice configuraţie de m (m>=4) bomboane plasate în n>=4 borcane există o secvenţă de mutări prin care putem plasa toate bomboanele într-un singur borcan”.
P(4)
Există 4 borcane care conţin cele 4 bomboane. Distribuţiile posibile (în diferite cazuri) în borcane sunt:
1. (1,1,1,1)
2. (1,3,0,0), evident pot fi şi (0,1,0,3), ...
3. (2,2,0,0) , evident pot fi şi (0,2,0,2), ...
4. (1,2,1,0) , evident pot fi şi (1,1,0,2), ...
Cazul (1) poate fi rezolvat, de exemplu, astfel:
(1,1,1,1)->(3,1,0,0)->(2,0,2,0)->(1,0,1,2)->(0,0,0,4)
Se observă că toate celelalte distribuţii apar în această secvenţă de mutări, deci îProbleme rezolvate oanele pot fi puse în acelaşi borcan.
Să presupunem acum ca **P(m)** este adevărată.
Să demonstrăm **P(m+1)**.
Vom alege o bomboană oarecare şi o vom marca (s-o numim bomboana specială). Din ipoteza inductivă deducem că toate celelalte bomboane pot fi plasate printr-o secvenţă de mutări într-un singur borcan.

a. Dacă acest borcan conține bomboana specială - am rezolvat problema.

b. În caz contrar, alegem două borcane goale oarecare şi se poate proceda, de exemplu, astfel:

`(1,m,0,0)->(0,m-1,2,0)->(0,m-2,1,2)->(2,m-3,0,2)->(1,m-1,0,1)->(0,m+1,0,0)`

Acum toate bomboanele sunt într-un singur borcan.
Algoritmul de determinare a secvenței de mutări se poate deduce din demonstrație: se aduc toate bomboanele în `4` borcane oarecare (pentru comoditate se aleg primele `4`). Se disting aici `4` cazuri:

1. `n=4`, se trece la final
2. `n=5`, se alege câte o bomboană din borcanul `5` şi din borcanul care are maximul de bomboane dintre primele `4`, şi se mută în borcanul care are minimul de bomboane dintre primele `4`
3. `n=6`, se alege câte o bomboană din borcanele `5` şi `6` şi se mută în borcanul care are minimul de bomboane dintre primele `4`; acest lucru se face până când se goleşte un borcan; acum suntem în cazul `2`, …
4. `n>6`, se alege câte o bomboană din borcanele cu cele mai multe bomboane dintre borcanele `5, 6, …, n` şi se mută în borcanul care are minimul de bomboane dintre primele `4`; acest lucru se face până când se ajunge la cazul `2`, …

Odată ajunse toate bomboanele în primele `4` borcane, le aducem la forma din demonstrație mutând de fiecare dată din borcanele care au maxim de bomboane (dintre borcanele `1`, `2`, `3`) în borcanul `4`, de exemplu, apoi, dacă este cazul, se urmează paşii din demonstrație.

Probleme rezolvate

Implementare în limbajul Pascal:

```
Program Borcane;
```

```
CONST MAXB = 100;
Type Borcan = Array[1..MAXB] Of LongInt;

Var Fin, Fout: Text;
    n: Integer;
    B: Borcan;
Procedure Citire;
Var i: Integer;
Begin
  Assign(Fin, 'bomboane.in'); Reset(Fin);
  ReadLn(Fin, n);
  For  i := 1 To n Do Read(Fin,B[i])
End;

Procedure Muta(x, y, z: Integer);
Begin
  Dec(B[x]); Dec(B[y]); Inc(B[z], 2);
  WriteLn(Fout, x, ' ', y, ' ', z)
End;

Procedure Fa4;
Var i, DeUnde: Integer;

Function Max: Integer;
Var i, M: Integer;
Begin
  M := B[4]; Max := 4;
  For i := 3 DownTo 1 Do
    If B[i]>M Then
      Begin
        M := B[i]; Max := i
      End;
End;

Function Min12(Var PM1: Integer): Integer;
Var i, M1, M2, PM2: Integer;
Begin
    If B[1]<B[2] Then
      Begin
        M1 := B[1]; PM1 := 1;
        M2 := B[2]; PM2 := 2;
      End
    Else
      Begin
```



```
      End;
  For i := 3 To 4 Do
      If B[i]<M1 Then
        Begin
          M2 := M1; PM2 := PM1;
          M1 := B[i]; PM1 := i
        End
      Else
```

```
        If B[i]<M2 Then
          Begin
            M2 := B[i]; PM2 := i
          End;
   Min12 := PM2
End;

Procedure Fa54(k: Integer);
Var Ma, Mi1, Mi2: Integer;
Begin  {in borcanul k mai sunt bomboane}
  While B[k]>0 Do
    Begin
      Ma := Max; Mi2 := Min12(Mi1);
      Muta(Ma, k, Mi1)
    End;
End;

Procedure Muta6;
Var Mi1, Mi2: Integer;
Begin
  While (B[5]>0) AND (B[6]>0) Do
    Begin
      Mi2 := Min12(Mi1);
      Muta(5, 6, Mi1)
    End;
  If B[5]>0 Then DeUnde := 5
  Else
    If B[6]>0 Then DeUnde := 6
    Else DeUnde := 0;
End;

Procedure Mutan_4;
Var i, Max1, Max2, Mi1, Mi2: Integer;

Function MaiSunt: Boolean;
Var Cate, i: Integer;
Begin
  Cate := 0;
  For i := 5 To n Do
    If B[i]>0 Then Inc(Cate);
  MaiSunt := Cate>1;
End;

Function Max(Var PM2: Integer): Integer;
Var i, M1, M2, PM1: Integer;
Begin
  If B[5]>B[6] Then
    Begin
      M1 := B[5]; PM1 := 5;
      M2 := B[6]; PM2 := 6
    End
  Else
    Begin
      M1 := B[6]; PM1 := 6;
```

```
      M2 := B[5]; PM2 := 5
    End;
  For i := 7 To n Do
    If B[i]>M1 Then
      Begin
        M2 := M1; PM2 := PM1;
        M1 := B[i]; PM1 := i
      End
    Else
      If B[i]>M2 Then
        Begin
          M2 := B[i]; PM2 := i
        End;
    Max := PM1
End;

Begin
  While MaiSunt Do
    Begin
      Max1 := Max(Max2);
      Mi2 := Min12(Mi1);
      Muta(Max1, Max2, Mi1)
    End;
  For i := 5 To n Do
    If B[i]>0 Then DeUnde := i;
End;

Begin
  If n=4 Then Exit
  Else
    If n=5 Then Fa54(5)
    Else
      If n=6 Then
        Begin
          Muta6;
          If DeUnde>0 Then Fa54(DeUnde);
        End
      Else
        Begin
          Mutan_4;
          If DeUnde>0 Then Fa54(DeUnde)
        End;
End;

Proc
Var i
    z1, z2: Integer;

Function Doua_Zero: Boolean;
Var Cate0, i: Integer;
Begin
  Cate0 := 0;
  For i := 1 To 3 Do
    If B[i]=0 Then Inc(Cate0);
```

```
  Doua_Zero := Cate0>=2;
End;

Function Max(Var M2: Integer;
             Var Min: Integer): Integer;
Var M1, PM1, PM2: Integer;
Begin
  If B[1]<B[2] Then
    Begin
      M1 := B[2]; PM1 := 2;
      M2 := B[1]; PM2 := 1;
      Min := 3
    End
  Else
    Begin
      M2 := B[1]; PM1 := 1;
      M1 := B[2]; PM2 := 2;
      Min := 3
    End;
  If B[3]>M1 Then
    Begin
      Min := PM2;
      M2 := M1; PM2 := PM1;
      M1 := B[3]; PM1 := 3
    End
  Else
    If B[3]>M2 Then
      Begin
        Min := PM2;
        M2 := B[3]; PM2 := 3
      End;
  M2 := PM2;
  Max := PM1
End;

Function Exact_2_Zero(Var Z1: Integer;
                      Var Z2: Integer): Boolean;
Var Cate0, i: Integer;
Begin
  Cate0 := 0;
  For i := 1 To 3 Do
    If B[i]=0 Then
      Begin
        Inc(Cate0);
        If Cate0=1 Then Z1 := i;
        If Cate0=2 Then Z2 := i;
      End;
  Exact_2_Zero := Cate0=2
End;
Begin
  While NOT Doua_Zero Do
    Begin
      Ma1 := Max(Ma2, Mi);
      Muta(Ma1, Ma2, 4)
```

```
    End;
  While Exact_2_Zero(Z1, Z2) Do
    Begin
      Muta(6-Z1-Z2, 4, Z1);
      Muta(Z1, 4, Z2);
      Muta(Z2, 4, 6-Z1-Z2);
      Muta(Z1, 6-Z1-Z2, 4);
      Muta(Z2, 6-Z1-Z2, 4);
    End;
End;

Begin
  Assign(Fout, 'bomboane.out');
  ReWrite(Fout);
  Citire;
  Fa4;
  Fa1;
  Close(Fout);
End.
```

Soluție 2 – (asistent Andreica Mugurel – Univesitatea Bucureşti)

```
#include <stdio.h>

#define NMAX 1010

int b[NMAX];
int i, j, b1, b2, dest, N, cnt;

int main()
{
    freopen("borcane.in", "r", stdin);
    cnt = 0;
    scanf("%d", &N);
    for (i = 1; i <= N; i++)
    {
      scanf("%d", &b[i]);
      if (b[i] > 0)
          cnt++;
    }
    freopen("borcane.out", "w", stdout);
    if (cnt <= 1) return 0;
    for (i = 1, j = 2; i <= N - 2; i++)
```



```
          j = i + 1;
      while (b[i] > 0)
      {
          while (b[j] == 0 && j < N)
              j++;
          if (j == N)
              break;
          printf("%d %d %d\n", i, j, N);
```

```
            b[i]--;
            b[j]--;
            b[N] += 2;
        }
        if (b[i] > 0)
            break;
    }
    for (i = 1; i < N; i++)
        if (b[i] > 0)
            break;
    if (i < N)
    {
        // a mai ramas un borcan (in afara de
       //ultimul) cu >0 bomboane (indicele i)
        if (i == 1)
            b1 = 2, b2 = 3;
        else
        if (i == N - 1)
            b1 = N - 3, b2 = N - 2;
        else
            b1 = i - 1, b2 = i + 1;
        dest = N;
        if (b[i] > b[dest])
        {
            j = i; i = dest; dest = j;
        }
        while (b[i] > 0 && b[dest] > 0)
        {
    if (b[i] == 1)
            {
                printf("%d %d %d\n",i,dest,b1);
                b[i]--; b[dest]--; b[b1] += 2;
                j = i; i = b1; b1 = j;
            }
            else
            {
                printf("%d %d %d\n",i,dest,b1);
                b[i]--; b[dest]--; b[b1] += 2;
                printf("%d %d %d\n",i,dest,b2);
                b[i]--; b[dest]--; b[b2] += 2;
                printf("%d %d %d\n%d %d %d\n",
                   b1, b2, dest, b1, b2, dest);
                b[b1]-=2; b[b2]-=2; b[dest]+=4;
            }
        }
    }
    return 0;
}
```


Intersecție

Într-o intersecție se găsesc NR maşini. Pentru simplificare vom considera că intersecția este o matrice NxM şi maşinile se află fiecare în câte o căsuță a acestei matrici. Fiecare maşină doreşte să ajungă la una din marginile matricii (doar la una) şi în acest scop se vor deplasa doar într-o singură direcție (direcția în care se află respectiva margine). O maşină nu poate trece printr-un patratel care este deja ocupat de o altă maşină. În momentul când maşina ajunge la marginea care o interesează ea dispare din intersecție.

Cerință

Dându-se configurația unei interesecții să se determine dacă pot ajunge toate maşinile la margine.

Date de intrare

În fişierul **intersectie.in** se vor găsi pe prima linie două numere N şi M. Pe următoarele N linii se vor găsi câte M numere cu semnificatia:

0 – patratel liber
1 – maşina care merge spre nord
2 – maşina care merge spre vest
3 – maşina care merge spre sud
4 – maşina care merge spre est

Date de intrare

În fişierul de ieşire **intersectie.out** se va găsi un singur număr, care reprezintă răspunsul la cerință:1 dacă pot părăsi intersecția **toate** maşinile şi 0 în caz contrar.

Restricții

1<=N,M<=50
1<=NR<=20 (numarul de maşini)

Exemple

intersectie.in	intersectie.out	explicații
3 3 3 0 2 0 0 0 Probleme propuse 4 0 1	1	Maşina 1,1 merge spre dus o pătratică. Apoi maşina din 1,3 iese din intersecție mergând spre vest până ce ajunge la marginea de vest. Analog ies şi celelalte maşini.
2 2 3 0	0	Maşinile 1,1 şi 2,1 nu pot să iasă din intersecție pentru că se blochează

1 1		reciproc.

Acoperire

Se dă o tablă de şah de dimensiuni NxM. Să se pună pe această tablă piesa de formă:

Piesele pot fi rotite cu 90, 180, 270 grade. Fiecare piesă acoperă exact trei pătrate din tablă.

Cerință

Să se determine numărul maxim de piese care se pot pune pe o tablă astfel încât acestea să nu se suprapună.

Date de intrare

Pe prima linie a fişierului **acoperire.in** se vor găsi 2 numere N şi M cu semnificația din enunț.

Date de ieşire

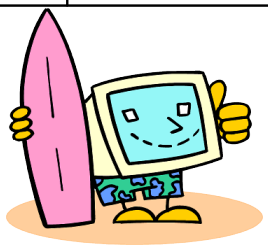
Fişierul **acoperire.out** conține o singură linie pe care se va găsi numarul maxim de piese.

Restricții

1<=N,M<=500

Exemple

acoperire.in	acoperire.out	explicații
3 3	2	Teoretic tabla are loc pentru 3 piese deoarece are 9 pătrățele. Dar observăm că oricum le-am pune pe primele 2 a treia nu mai are loc pe tablă



Paicu Alexandru, clasa a XI-a D