Bimestrale per l'insegnamento delle scienze e della matematica - Anno XXXII - n. 6 fascicoli all'anno - Direttore Responsabile: Luigi Morgano - Autorizzazione del Tribunale di Brescia n. 228 del 31 marzo 1965 - Sped. in abbonamento postale/45% - Comma 26, Art. 2 Legge 549/95 - Filiale di Brescia (Italia) - **Direzione, Redazione, Amministrazione:** La Scuola S.p.A. - 25186 Brescia, Via Luigi Cadorna, 11 - Conto corrente postale n. 14407258 - Codice fiscale - Partita I.V.A. n. 00272780172 - Tel. centr. (030) 2993.1 - Tel. Ufficio Abbonamenti (030) 2993.286 - (030) 2993.246 - Telefax (030) 2993.299. **Filiali: 40131 Bologna** (Via Cipriani, 5, Tel. (051) 521090 - 521180 - Telefax (051) 521090); **20136 Milano** (Viale Bligny, 7, Tel. (02) 58300261 - 8353716 - Telefax (02) 8353716); **00193 Roma** (Via Crescenzio, 23, Tel (06) 6875179 - 68803989 - Telefax (06) 6874939); **80137 Napoli** (Salita S. Elia, 19/21, Tel. (081) 441200 - 441308 - Telefax (081) 441934); **70124 Bari** (Via Giulio Petroni, 21 A/E - Tel. (080) 5428647 - Telefax (080) 5428647); **65124 Pescara** (Via Donatello, 7/11, Tel. (085) 74792 - Telefax (085) 74792); **35129 Padova** (Via della Croce Rossa, 116 - Tel. (049) 8076775 - 8076776 - Telefax (049) 8076776). **Pubblicità:** Ufficio Inserzioni Pubblicitarie Editrice La Scuola - Via Cadorna, 11 - 25186 Brescia - tel. (030) 2993.287 - Telefax (030) 2993.299. **Abbonamento annuo** 1996-97 L. 60.000 pagabile in un'unica soluzione (estero via terra/mare L. 68.000 - tariffe differenziate per via aerea) - Il presente fascicolo L. 10.000 (arretrato il doppio). **L'impegno di abbonamento è continuativo,** salvo regolare disdetta da notificarsi a mezzo lettera raccomandata. Stampa: OFFICINE GRAFICHE LA SCUOLA - 25186 BRESCIA.

DIDATTICA DELLE SCIENZE
e informatica nella scuola

187 gennaio 1997 - anno XXXII

direttore
Mauro Laeng, docente di Pedagogia all'Università di Roma

comitato di redazione
Giuseppe Luciano (caporedattore)
Angelo Borgese, Gemma Colosio, Mario Ferrari, Marina Gandellini
Teresita Giliani, Cecilia Mazzocchi, Laura Pasolini,
Pierluigi Pizzamiglio, Ennio Provera, Maria Pia Rigobello

In copertina

La lampada

(Foto: *Marka - Free Lance Associati 2*)

Didascalia a pagina 6.

Progetto Grafico di *Giuliano Prati*

Ad agni comunicazione o richiesta riguardante la rivista i sigg. abbonati sono pregati di allegare una copia del talloncino-indirizzo col quale la rivista stessa viene loro spedita.

3 *Mauro Laeng*
Editoriale

5
Per discutere: discutiamo

6 *Sergio Calzolani*
La lampada

8 *Raffaele Mongodi*
Le origini del modello meccanico

13 *Leonarda Calianno, Cinzia De Mita, Piermario Pedone, Domenico Potenz*
Protocollo per l'analisi della situazione d'ingresso nella scuola secondaria superiore

18 *Catia Pardini*
Insegnamento delle scienze con la didattica per concetti

25 *Cristina Mostosi*
Verifica sperimentale del contenuto lipidico di alcuni alimenti (metodo Soxhlet)

33 *Orazio Casella*
Studio del moto di un corpo su un piano inclinato con sonar on-line

39 *Iacopo Riani*
Energia ed ambiente, un equilibrio possibile

43 *Angelo Bressan*
Membrane elestiche con sistemi algebrici

47 *Mirella De Amici*
Introduzione all'uso di Derive. Scheda n. 5

54 Prova è facile
Marco Calvani
Il computer che si autoistruisce: un gioco in Pascal per cominciare

60
Asterischi

62
Notiziario

63
Recensioni

Prova: è facile

IL COMPUTER CHE SI AUTOISTRUISCE: UN GIOCO IN PASCAL PER COMINCIARE

«L'attività dell'immaginazione dipende dalla quantità e dalla qualità del materiale disponibile. Nulla c'è nell'immaginazione che prima non sia stato nel senso».

Montgomery Belgian [1]

Presentazione

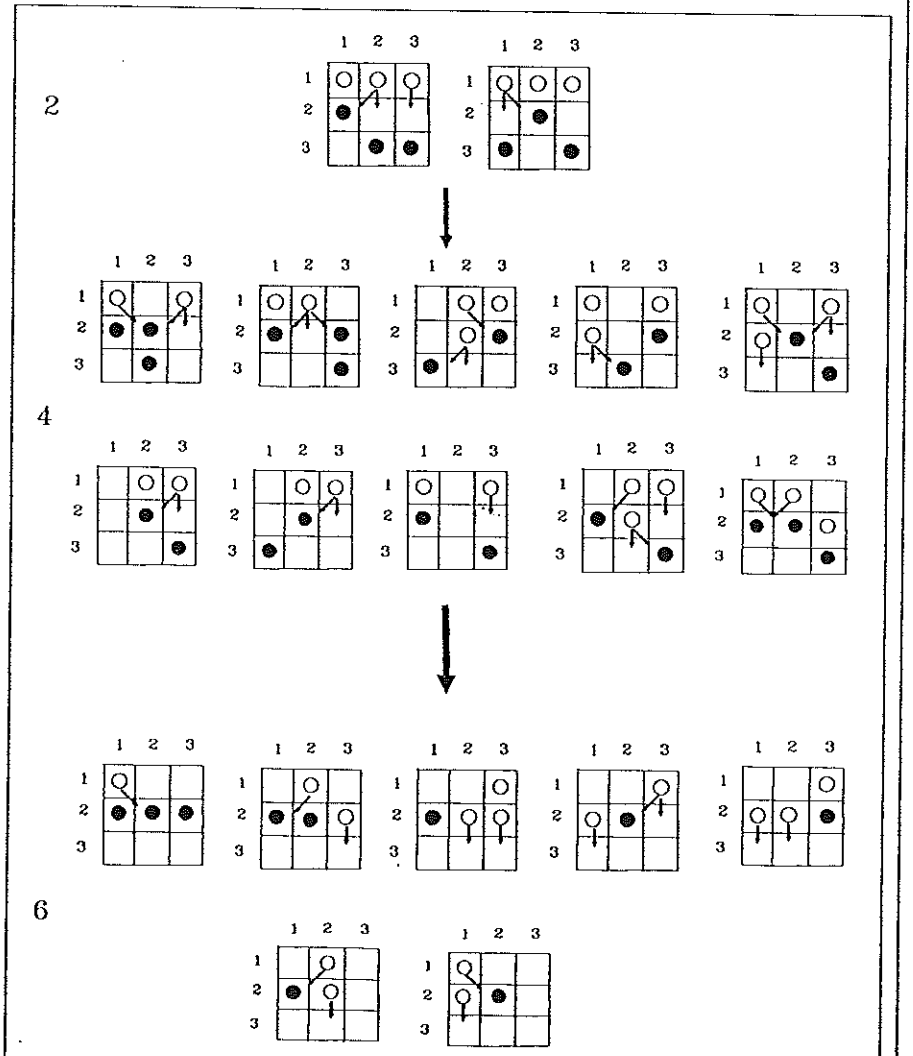
L'articolo nasce con l'intento di proporre agli studenti l'argomento relativamente recente[1] dell'autoistruzione; le prime scoperte a questo riguardo hanno inizio nel 1959 con Arthur L. Samuel che programmò un calcolatore in modo che potesse giocare a dama facendo tesoro delle partite fatte e migliorando la propria strategia di gioco, partita per partita, fino a diventare un ottimo giocatore, con la differenza sostanziale rispetto a un giocatore di «scarsa memoria», di non commettere mai lo stesso errore [2].
I giochi si classificano in:
1) giochi di pura fortuna;
2) giochi di pura abilità;
3) giochi di abilità e fortuna.
I giochi che vengono utilizzati negli esperimenti dell'autoistruzione sono quelli di pura abilità e con due giocatori (giochi a due).
Tali giochi, nel caso in cui non sia previsto il pareggio, ammettono una strategia vincente, cioè esiste un insieme di mosse da effettuare, per uno dei due giocatori, che fanno vincere. Il problema matematico consiste nel determinare l'eventuale strategia vincente e scoprire per quale dei due giocatori è valida.
Questo tipo di problemi trova diverse applicazioni nel reale: in campo militare per lo studio delle strategie di guerra (ricordate il computer della base militare nel film *War Games*?), in campo civile nella crittologia per la ricerca, non autorizzata, della chiave di lettura di documenti codificati.
Nell'articolo si propone di costruire insieme agli studenti, utilizzando il linguaggio Pascal, l'*HER (Hexpawn Educable Robot* = robot educabile per esapedone), una macchina che si autoistruisce progettata per il gioco dell'esapedone, ideato da Martin Gardner [2].
Il gioco dell'**esapedone** essendo di pura abilità, ammette una strategia vincente e se tutti e due i giocatori ne sono a conoscenza vince chi muove per secondo. Quindi per far determinare la strategia al computer sarà necessario farlo giocare sempre per secondo: la strategia vincente è facilmente determinabile e, proprio per la sua immediata risolubilità, rappresenta per lo studente lo strumento più adatto per l'inserimento in questo settore della conoscenza matematica.
Il programma proposto utilizza gli strumenti base del Pascal e può essere, quindi, realizzato anche con qualsiasi versione del linguaggio stesso.

Il gioco e le regole

L'esapedone si gioca su una scacchiera 3×3; la configurazione iniziale si ottiene disponendo tre pedoni bianchi su di un lato della scacchiera e tre sul lato opposto, come in figura 1. I pedoni si possono muovere secondo le seguenti regole:

1) un pedone può procedere in avanti da un quadrato ad un altro quadrato vuoto;

2) un pedone può catturare un pedone nemico muovendosi di un quadrato in diagonale, a sinistra o a destra, su una casella occupata dall'avversario;



2,4,6 sono gli schemi che si presentano al computer per la scelta della mossa risp. 2,4,6 del gioco.

fig. 1

3) un pezzo catturato viene rimosso dalla scacchiera.

Il gioco si vince:

a) facendo avanzare il pedone sino alla riga di partenza dell'avversario;
b) catturando tutti i pezzi nemici;
c) raggiungendo una posizione grazie alla quale l'avversario non può più muovere.

Robert A. Ellis, del laboratorio di Calcolo dei *Balistics Research Laboratories*, al poligono sperimentale di Arbedeen nel Maryland, è il primo a parlare di programma digitale che applica la tecnica dell'autoapprendimento [2].

Il processo di insegnamento

Per chiarire i termini che verranno utilizzati in seguito diamo le seguenti definizioni:

1) *configurazione* è una rappresentazione delle posizioni dei pedoni sulla scacchiera;

2) *schema* è una coppia costituita da una configurazione e delle mosse possibili relative.

L'utente fa la prima mossa; il computer, nel quale sono stati inseriti tutti gli schemi di gioco possibili che gli possono capitare, riconosce la mossa fatta dallo sfidante tramite l'individuazione, negli schemi in memoria, della configurazione corrispondente e sceglie una tra le sue mosse possibili dallo schema selezionato. Si procede di seguito secondo le regole fino alla fine della prima partita in cui il computer perde. Il computer, quando ha perso, si autopunisce eliminando la sua *ultima* mossa dalla memoria. Nelle partite successive, se il computer giunge ad uno schema per il quale nella sua memoria non ha mosse da fare, abbandona e si autopunisce confiscandosi l'ultima mossa precedente.
Dopo aver perso 11 partite il computer diventa un giocatore perfetto!!

Suggerimenti didattici

Si suggerisce di far costruire prima la macchina (scrivere i programmi) e imparare il gioco contemporaneamente ad essa (è proprio vero, "giocando s'impara")[2].
Il programma ed il gioco si prestano a diverse modifiche:

– si può pensare di premiare il computer ogni volta che vince favo-

I Fase, registrazione di tutte le mosse possibili

```
PROGRAM HER_I (input,inf);

type      {FASE I inserimento schemi e mosse possibili, su file fisico}

 schema = record                          dichiarazione del record che viene denominato
                                          schema; le componenti
 pos : array [1..3,1..3] of integer;      sono: una matrice di interi 3x3 per il modello, una
                                          stringa di 17 caratteri
    mosse : string[17];                   per le mosse possibili
 end;

 type spettro = array [1..3,1..3] of integer;   dichiarazione della matrice che viene chiamata
                                          spettro per lo studio dei
 var a:spettro;                           vari modelli: a è uno spettro
    ss:string[8];                         ss è una stringa di 10 caratteri
 schemacorr:schema;                       schemacorr è un record del tipo schema
    inf : file of schema;                 inf è un file di records di tipo schema
    i,j,k,z: integer;
    fine: boolean;
    risp: char;                           risp è una variabile di tipo char utile per la scelta
                                          dell'utente

 BEGIN                                    inizio del programma principale della I fase
 ss:='tuttemos';
 assign(inf,ss);
 rewrite(inf);                            apertura file tuttemos
 for k:=0 to 18 do
 begin
    writeln('Schema n.', k+1);
    for i:=1 to 3 do
      for j:=1 to 3 do
        begin
          write('In a[',i,',',j,']  che c''è ? ');
          readln(schemacorr.pos[i,j])
        end;
    writeln('INSERISCI TUTTE LE MOSSE E PREMI INVIO');
    readln(schemacorr.mosse);
    write(inf,schemacorr)                 scrittura dei dati su file
  end;
 close(inf)                               chisura del file tuttemos
 END.                                     fine del programma principale della I fase
```

II Fase, costruzione del gioco

```
PROGRAM HER_II(input,output,inf);
 uses crt;                                uso della unit crt contenente funzioni per l'utilizzo
                                          dello schermo per es. clscr (pulisce lo schermo in
    type      {FASE II}                   out-put)
  schema = record
 pos : array [1..3,1..3] of integer;
    mosse : string[17];
 end;

 type spettro = array [1..3,1..3] of integer;
 var a:spettro;
    ss:string[8];                         ss è una stringa di 8 caratteri
    ammessa,uguali,ugualisim,vittmia,vittsua: boolean;  variabili booleane per controlli vari
    schemacorr:schema;
    inf : file of schema;
    i,j,pr,pc,sr,sc,z,count,l,m,memos,memom: integer;
    risp: char;                           risp è una variabile di tipo char utile per la scelta
                                          dell'utente di chiudere il gioco

 procedure schermo(b:spettro);            schermo(modello) mostra in out-put il modello
    var e,f: integer;                     corrente della partita

  begin
  clrscr;
  writeln;writeln;writeln;
  for e:=1 to 3 do
    begin
    for f:=1 to 3 do
```

rendo la strategia vincente (un modo può essere aumentare la probabilità di scelta delle mosse vincenti);

– modificare il numero dei pedoni in gioco e di conseguenza la scacchiera;

– far ricavare al computer stesso gli schemi e le mosse possibili;

– far giocare il computer contro se stesso in maniera tale che si autoistruisca autonomamente;

– si accettano altri suggerimenti.

Costruire l'HER

Si è pensato di rappresentare la scacchiera con una matrice di numeri interi 3×3: lo zero corrisponde ad una casella vuota, l'uno è una pedina del computer, il due è una pedina dell'utente.
Ci si muove sulla scacchiera indicando le coordinate matriciali di partenza della pedina che si intende muovere ed inserendo successivamente quelle di arrivo.
Tutti gli schemi possibili, le configurazioni della scacchiera con le relative mosse, nei quali può «incappare» il computer durante la partita (vedi fig. 1) vengono introdotti nel computer dal programmatore.
Ci si muove sulla scacchiera indicando le coordinate matriciali di partenza della pedina che si intende muovere ed inserendo successivamente quelle di arrivo.
I **tipi** utilizzati sono:

– *integer*, *char*, *boolean* più comunemente usati nelle scuole superiori;

– la stringa di caratteri, *string* [lunghezza in caratteri della stringa], usata, ad esempio, per memorizzare tutte le mosse per ogni schema (lung. 17). Tale stringa raccoglie in ordine sequenziale le coordinate di partenza e quelle di arrivo relative a ciascuno schema, l'ultimo carattere della stringa è indicato con la lettera *f*;

– il *record*, è una rappresentazione dello schema e quindi è una variabile che contiene una matrice 3×3 ed una stringa di 17 caratteri; l'utilità del *record* sta nella possibilità di associare ad ogni schema tutte le mosse possibili;

– il tipo *file of* nome del *record* da inserire, file di dati che memorizza tutti gli schemi possibili (il nome del file, poi scritto sul disco, viene assegnato tramite una stringa di 8 caratteri).

```
    write('    ',b[e,f],'    ');
    writeln;
  end;
end;

procedure ammissione;

begin
 if (pr-1>=0) then
    begin
     if (pc=sc) and (sr=(pr-1)) then
        if (a[sr,sc]=0) then
           ammessa:=true;
     if (sc-1>=0) then
          if (a[sr,sc]=1) and (pc=sc-1) then
            if (sr=pr-1) then ammessa:=true;
     if (a[sr,sc]=1) and (pc=sc+1) then
          if (sr=pr-1) then ammessa:=true
    end;
end;
```

ammissione verifica l'ammissibilità della mossa fatta dall'utente secondo le regole del gioco, si ricorda che le mosse vengono memorizzate come coordinate matriciali cioè con tipi interi

```
procedure sua_mossa;

 begin
  ammessa:=false;
  repeat
  writeln;
  writeln('  ------------------------------------------------------');
  writeln(' TOCCA A TE!! INTRODUCI LE COORDINATE MATRICIALI DELLA  PEDINA');
  writeln;
  write('  DA MUOVERE ..... Pr =');read(pr);
  write('                Pc =');readln(pc); writeln;
  write('  LA SPOSTI IN ... Sr =');read(sr);
  write('                Sc =');readln(sc);
  writeln;
  ammissione;
  if (not ammessa) then
          begin
           writeln('MOSSA NON AMMESSA (ripassa le regole) RIPROVA !');
           delay(2000)
          end;
 until ammessa;
  a[pr,pc]:=0;
  a[sr,sc]:=2
end;
```

sua_mossa serve a colloquiare con l'utente e fargli fare la mossa, la richiesta della mossa viene ripetuta (*repeat until ....*) fino a quando non è ammessa, quindi viene modificato il modello corrente (*a*) secondo la richiesta

```
procedure vitt(var vittoria:boolean; h,l,m:integer);
 var e,f: integer;

  begin
   vittoria:=true;
   for e:=1 to 3 do
     for f:=1 to 3 do if (a[e,f]=h) then vittoria:=false;
   for e:=1 to 3 do if (a[m,e]=l) then vittoria:=true;
  end;
```

vitt(....) permette di verificare se il giocatore richiesto (*l*) verifica 2.a (*m* numero della riga da controllare), o se viene verificata 2.b (*h* è l'altro giocatore)

```
procedure bloc(var vittoria:boolean);
 var e,f: integer;
    patta: boolean;

 begin
  patta:=true;
  for f:=1 to 2 do
    for e:=1 to 3 do
      if (a[f,e]=1) and (a[f+1,e]<>2) then patta:=false;
  for f:=3 downto 2 do
    for e:=1 to 3 do
      if (a[f,e]=2) and (a[f-1,e]<>1) then patta:=false;
  for f:=1 to 2  do
    for e:=1 to 2 do
      if (a[f,e]=1) and (a[f+1,e+1]=2) then patta:=false;
  for f:=1 to 2 do
    for e:=2 to 3 do
      if (a[f,e]=1) and (a[f+1,e-1]=2) then patta:=false;
    if patta then vittoria:=true
 end;
```

bloc(....) verifica l'eventuale vittoria raggiunta con una patta (2.c)

In figura 1 sono mostrati tutti gli schemi possibili; per abbreviare vengono trascurati alcuni modelli simmetrici; questi sono analizzati nel programma tramite «sottoprogrammi» (procedure) che sfruttano le condizioni di simmetria delle matrici.

Esempio
Dovendo registrare il primo modello della seconda mossa di figura 1, vengono inseriti: una matrice 3×3 con i seguenti dati $\begin{matrix} 1 & 1 & 1 \\ 2 & 0 & 0 \\ 0 & 2 & 2 \end{matrix}$ e la stringa 122112221323f.

```
procedure trovaschema;
  var e,f:integer;

 begin
  z:=0;
  reset(inf);
  repeat
  seek(inf,z);
  read(inf,schemacorr);
  uguali:=true;
  for e:=1 to 3 do
   for f:=1 to 3 do if (a[e,f]<>schemacorr.pos[e,f]) then uguali:=false;
  z:=z+1
  until uguali or (z=filesize(inf));
  if (z=filesize(inf)) then z:=memos+1;
  z:=z-1;
  close(inf)
 end;
```

trovaschema, applicato dopo la mossa dell'utente, ricerca il modello corrente nel file di dati e legge lo schema corrispondente

apre il file di dati *inf*

posiziona il cursore sullo schema con indice *z*
legge lo schema dal file

chiude il file di dati

```
procedure trovaschemasim;

   var e,f:integer;

 begin
  z:=0;
  reset(inf);
  repeat
  seek(inf,z);
  read(inf,schemacorr);
  ugualisim:=true;
  for e:=1 to 3 do
   for f:=1 to 3 do if (a[e,4-f]<>schemacorr.pos[e,f]) then ugualisim:=false;
  z:=z+1
 until ugualisim or (z=filesize(inf));
 if (z=filesize(inf)) then z:=memos+1;
 z:=z-1;
 close(inf)
end;
```

trovaschemasim viene utilizzato per verificare la possibilità che il modello corrente sia simmetrico a quello memorizzato

```
function decoder(a: char):integer;

  begin
   case a of
    '1': decoder:=1;
    '2': decoder:=2;
    '3': decoder:=3
   end;
  end;
```

decoder(..) è una funzione che trasforma il carattere (dalla stringa delle mosse) nel numero corrispondente

```
function esistenza(a: string):boolean;
  var i:integer;
 begin
  i:=1; esistenza:=false;
  repeat i:=i+1
  until a[i]<>'n';
  if a[i]<>'f' then esistenza:=true
 end;
```

esistenza(..) verifica se lo schema in questione ammette mosse possibili e segnala il risultato del controllo con un tipo booleano

(segue a pag. 58 e 59)

Il programma che effettua l'inserimento di tutti i dati ha nome *HER_I*, mentre il gioco viene gestito dal programma *HER_II*.
Con *HER_I* (fig. 2) viene generato il file *TUTTEMOS*[3] su disco (drive di *default*), in modo tale che quando si passerà alla seconda fase (il gioco) il computer, al momento di muovere, andrà a cercare lo schema al quale fare riferimento e sceglierà la mossa.
Per una migliore comprensione dei programmi, scritti in corsivo sul lato destro del testo che segue, si è pensato di illustrare le varie procedure e funzioni; i commenti sono in caratteri normali.

Marco Calvani
Liceo Scientifico «P. Paschini»
Gemona (Udine)

[1] «Occorre che la scuola per continuare ad essere alla base del progresso sociale in un'era di così rapide trasformazioni trovi il mezzo di ringiovanire e trasformare l'istruzione che offre, introducendo nei suoi programmi le nuove scoperte del nostro tempo. Tutto ciò dipende alla fin fine dalla capacità di coltivare e dare espressione ai modelli di perfezione che emergono dalla nostra poliedrica civiltà». Mazzetti, Dewey e Bruner, *Il processo educativo nella società industriale*, 1976, Armando ed., Roma.
[2] «Se il ragazzo non si appropria del problema, cioè non si pone come obiettivo di cercare risposte a certe domande, non nasce la motivazione al lavoro». Pincella, Malara, *Lo studio informale delle trasformazioni e degli invarianti come approccio alla geometria nella scuola media*, La matematica e la sua didattica, 1995, n. 4, pp. 446-462.
[3] Si suggerisce, una volta creato il file su disco, di salvarlo con altro nome (per esempio con MS-DOS *copy tuttemos tuttecop*) in maniera tale da poterlo riutilizzare dovendo far ripartire il gioco, visto che le modifiche che vengono apportate al file durante il gioco stesso modificano il file anche su disco.

Bibliografia

[1] J.D. Barrow, *La luna nel pozzo cosmico*, 1992, Adelphi, Milano.
[2] Martin Gardner, *Enigmi e giochi matematici*, 1975, vol. 4, Sansoni, Firenze.

```
procedure mia_mossa;                          mia_mossa permette al computer di elaborare
                                              la sua mossa
  begin
   l:=1;
   trovaschema;
   if (not uguali) then trovaschemasim;
       if esistenza(schemacorr.mosse) then
     begin
      i:=1;
      repeat l:=l+4 until schemacorr.mosse[l]='f';   calcola il numero delle mosse possibili relative
                                                     allo schema trovato

repeat m:=random((l-i)div(4)) until (schemacorr.mosse[i+m*4] <>'n'); scelta casuale della mossa
i:=i+m*4
       if uguali then
         begin
           a[decoder(schemacorr.mosse[i]),decoder(schemacorr.mosse[i+1])]:=0;
           a[decoder(schemacorr.mosse[i+2]),decoder(schemacorr.mosse[i+3])]:=1;
         end;
       if ugualisim then
         begin
           a[decoder(schemacorr.mosse[i]),4-decoder(schemacorr.mosse[i+1])]:=0;
           a[decoder(schemacorr.mosse[i+2]),4-decoder(schemacorr.mosse[i+3])]:=1
         end;
        memos:=z;                                 memorizza l'indice di file relativo allo schema
        memom:=i;                                 della mossa (memos) e l'indice di stringa della
      end;                                        mossa fatta (memom)
    end;


procedure elimina_mossa;                      elimina_mossa, in caso di sconfitta da parte del
  var e,k:integer;                            computer viene eliminata l'ultima mossa fatta
                                              relativa all'ultimo schema
  begin
   reset(inf);                                apre il file tuttemos
   seek(inf,memos);                           sceglie lo schema precedente alla sconfitta
   read(inf,schemacorr);
   for k:=memom to (memom+3) do schemacorr.mosse[k]:='n';   modifica le mosse
   seek(inf,memos);
   write(inf,schemacorr);                     modifica lo schema sul file
   close(inf)                                 chiude il file
  end;



BEGIN                                         programma Principale
 SS:='TUTTEMOS';
 ASSIGN(INF,'TUTTEMOS') ;
 REPEAT
  COUNT:=1;
  FOR i:=1 TO 3 DO a[1,i]:=1;                 inizializza il modello di partenza
  FOR i:=1 TO 3 DO a[2,i]:=0;
  FOR i:=1 TO 3 DO a[3,i]:=2;
  REPEAT
   SCHERMO(a);
   SUA_MOSSA;
   VITT(vittsua,1,2,1);
   BLOC(vittsua);
   IF NOT VITTSUA THEN
            BEGIN
             MIA_MOSSA;
             IF ESISTENZA(schemacorr.mosse) THEN
                        BEGIN
                          VITT(vittmia,2,1,3);


                          BLOC(vittmia)
                        END
                              ELSE
                     VITTSUA:=TRUE
            END;
      CLRSCR;
```

```
    UNTIL VITTMIA OR VITTSUA;
    IF VITTSUA THEN
        BEGIN
          SCHERMO(a);
          ELIMINA_MOSSA;
          WRITE('HAI VINTO, BRAVO !! MI FAI RIPROVARE (S,N)? ');
          READLN(risp)
        END
          ELSE
        BEGIN
          CLRSCR;
          SCHERMO(a);
          WRITELN('  HO VINTO IO SONO UN CAPO!!!');
          DELAY(3000);
          WRITE('  VUOI RITENTARE PIVELLO (S,N)? ');
          READLN(risp)
        END
  UNTIL (risp='N')or(risp='n')
END.
```


CORSO DI FISICA

metodi, concetti, esperimenti

per i bienni che adottano il Piano Nazionale per l'introduzione dell'Informatica e per il biennio degli I.T.I.

di **IACOPO RIANI**

cod. 8965, pp. 512, illustrato a col., L. 44.500

Il testo si rivolge agli studenti del biennio che adotta il Piano Nazionale per l'introduzione dell'Informatica, ma, data l'estrema flessibilità delle parti che lo compongono, il testo si presta ad essere utilizzato in tutte le situazioni in cui l'insegnamento della fisica prevede l'utilizzo del laboratorio e la possibilità di accedere alle strutture di un'aula informatica.

DISCHETTO OPERATIVO

(3,5" 720 Kbyte) Sistema MS-DOS compatibile
cod. 56625 FDP 625, L. 6200

FISICA. MODELLO DELLA REALTÀ

per i bienni che adottano il Piano Nazionale per l'introduzione dell'Informatica e per il biennio degli I.T.I.

di **IACOPO RIANI**

1. cod. 8271, pp. 256, illustrato a col., L. 27.000
2. cod. 8273, pp. 256, illustrato a col., L. 27.000

Anche questo testo si rivolge agli studenti del biennio Piano Nazionale per l'introduzione dell'Informatica, e grazie alla sua articolazione in due volumi, risulta altrettanto indicato per i corsi tradizionali, ove l'insegnante intenda procedere ad uno «svecchiamento» dei programmi, ad una introduzione «morbida» delle nuove tecnologie.

DISCHETTO OPERATIVO

(5,25" 360 Kbyte) Sistema MS-DOS compatibile
cod. 56608 FDP 608, L. 4150

DUE EDIZIONI PARALLELE
- per lo studente
- per l'insegnante

ADOZIONI 1997-98

EDITRICE LA SCUOLA