MDL – Modul de dezvoltare a lecţiilor asistate de calculator

Daniel Safta
Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică şi Calculatoare
Str. G.Bariţiu nr. 26-28
davonkeep@yahoo.com

Dorian Gorgan
Universitatea Tehnică din Cluj-Napoca,
Facultatea de Automatică şi Calculatoare
Str. G.Bariţiu nr. 26-28
dorian.gorgan@cs.utcluj.ro

REZUMAT
În acest articol vom prezenta o nouă abordare în didactica informaticii - modelarea vizuală a algoritmilor şi evaluarea bazată pe forme metaforice - aplicată în nucleul unui sistem de învăţământ virtual, un modul de dezvoltare a lecţiilor asistate de calculator (MDL). Am evidenţiat structura şi caracteristicile procesului didactic implementat în sistemul propus, utilizatori şi rolurile lor, metodele didactice aplicate, soluţii de evaluare şi studiu de caz pe un model de lecţie. Am expus stadiul actual în domeniul temei alese, evidenţiind avantajele soluţiei expuse, dar şi extinderi posibile.

Cuvinte cheie
Instruire asistată de calculator, modelare vizuală, proces didactic, gândire algoritmică, limbaj de programare.

Clasificare ACM
K. Computing Milieux, K.3 COMPUTERS AND EDUCATION, K.3.1 Computer Uses in Education, *Computer-assisted instruction (CAI).*

INTRODUCERE

La ora actuală există o puternică preocupare în domeniul didacticii informaticii spre introducerea utilizatorilor în ştiinţa calculatoarelor de la o vârstă cât mai fragedă. În acest sens, sunt binecunoscute eforturile Massachusetts Institute of Technology, care prin Lifelong Kindergarten [1] elaborează şi întreţin proiecte educaţionale îndreptate spre copii şcolari şi preşcolari. Motto-ul lor "Sowing the seeds for a more creative society" indică importanţa dezvoltării înclinaţiilor copiilor spre programare încă de la clasele primare. Trebuie realizată, pe lângă alfabetizarea propriu-zisă, o aşa numită alfabetizare în utilizarea calculatorului. În acest scop se utilizează programarea vizuală şi programarea prin exemple, implicarea copiilor în proiecte educative, cum sunt Scratch sau Intel Computer Clubhouse Network.

Scratch [2] se adresează copiilor cu vârste cuprinse între 6-16 şi încurajează trei direcţii: imaginează, programează, împarte cu ceilalţi. Iată cum un limbaj de programare este mascat într-un context educativ-distractiv, un mediu în care utilizatorii creează în mod interactiv şi împart cu restul comunităţii online propriile povestiri, jocuri, muzica preferată, artă etc. La ora actuală există mii de proiecte active pe site-ul Scratch iar seriozitatea proiectului este întărită de susţinători precum National Science Foundation, Microsoft, Intel Foundation, Nokia, Iomega şi consorţiul de cercetare de la MIT Media Lab.

Alte medii de promovare a informaticii în rândurile tinerilor sunt aşa numitele Computer Clubhouses precum Intel Computer Clubhouse Network [3], Lego Mindstorms [4], PicoCrickets [5] etc. În aceste comunităţi online utilizatorii împart propriile idei şi experienţe din domeniul programării vizuale elementare aplicate pe domenii practice cum ar fi reţelele de comunicaţii, roboţei lego programabili şi chiar artă (muzică, sculptură, dans, ateliere de bijuterie etc).

Pornind de la premisa că sunt deja familiarizaţi cu utilizarea calculatoarelor şi posedă minime aptitudini de programare, tinerii liceeni îşi pot dezvolta gândirea şi stilul de programare cu ajutorul unor sisteme bazate pe programarea vizuală precum Visual Basic Game Programming for Teens [6], Macromedia Authorware [7].

Cu toate acestea, familiarizarea tinerilor cu mediul informatic nu implică faptul că aceştia vor şti să dezvolte un algoritm sau că stăpânesc un limbaj de programare. Între statutul de utilizator şi cel de programator există o barieră foarte înaltă [8]. Există numeroase tentative de a ajuta utilizatorul în drumul său însă majoritatea întâlnesc aceeaşi piedica, stabilirea clară în gândirea umană a conexiunii între modelarea vizuală a soluţiei şi descrierea într-un limbaj de programare [9]. În special utilizatorii foarte tineri, învăţaţi cu acel mod introductiv de "programare distractivă" au deficienţe în adaptarea la un nou mod de abordare a algoritmilor, la abstractizarea noţiunilor modelate vizual şi transpunerea în cuvinte. Este nevoie de un anumit nivel de inteligenţă şi maturitate a gândirii individului pentru a putea face acest salt.

Motivaţie

În didactica informaticii, în interacţiunea dintre utilizator şi limbajul de programare, în special în cazul începătorilor, apare o mare deficienţă – modul de abordare. Utilizatorul începător nu are gândirea suficient structurată ca să elaboreze un algoritm constrâns de regulile unui pseudocod sau chiar ale unui limbaj de programare, totodată are deficienţe în descrierea traseului logic din rezolvarea unei probleme. Rezultă două mari necesităţi simultane: nevoia de a-şi dezvolta gândirea logică, capabilă de a abstractiza şi a transpune ideile într-un algoritm şi nevoia de a stăpâni foarte bine rigorile unui limbaj de programare.

Abordarea actuală la nivelul didacticii informaticii, în special la noi în ţară, este aceea că utilizatorul trebuie să-şi formeze gândirea astfel încât "să gândească precum calculatorul", ceea ce este o mare greşeală. În fond, algoritmii sunt realizaţi după gândirea umană şi de aceea vom construi întreg sistemul descris mai departe pe ideea că: utilizatorul scrie codul sursă prin acţiuni şi nu prin instrucţiuni!

MDL – MODUL DE DEZVOLTARE A LECŢIILOR ASISTATE DE CALCULATOR

Modulul de dezvoltare a lecţiilor asistate de calculator este nucleul unui întreg sistem de învăţământ virtual aflat în curs de dezvoltare, sistem accesibil atât local cât şi la distanţă, şi care oferă o alternativă de abordare a informaticii, în special a studiului algoritmilor şi limbajelor de programare. MDL furnizează lecţii pe baza unui nou concept în didactica informaticii, axat pe modul de abordare a utilizatorului, de interacţiunea sa cu mediul de programare printr-o interfaţă orientată spre modelarea vizuală şi interactivă a algoritmului, şi nu spre simpla editare de cod sursă, cea din urma realizându-se automat.

Noutatea constă în strategia didactică folosită în dezvoltarea pachetului de lecţii, în modul de evaluare şi de interpretare a rezultatelor: se va urmări forma finală a modelului descris şi nu paşii intermediari. Astfel:

- elevul nu este îngrădit în alegerea unei anumite soluţii, ci este liber să creeze propria cale de rezolvare a problemei, să modeleze vizual algoritmul după propriile idei;
- nu se pune accentul pe diagrama stărilor parcurse, ci pe rezultatul obţinut;
- abia după validarea soluţiei se va trece la analiza paşilor efectuaţi;
- dacă răspunsul e corect, se reţine traseul acţiunilor descrise ca o nouă metodă în rezolvarea problemei;
- dacă răspunsul nu coincide cu forma metaforică aşteptată, sistemul urmăreşte firul desfăşurării evenimentelor şi indică prima abatere întâlnită de la traseele cunoscute.

Acest concept stă la baza dezvoltării unui pachet de lecţii cu aplicare practică în studiul algoritmilor şi a tehnicilor de programare, particularizat pe nivele de vârstă, capacitate de înţelegere şi nu în ultimul rând pe baza cunoştinţelor anterioare.

Procesul de învăţământ

Combinând avantajele programării vizuale cu principiile programării structurate, se evidenţiază următoarea alternativă în interacţiunea utilizator – algoritm: utilizatorul nu este forţat, în prima etapă, spre a folosi un limbaj de programare, ci spre a modela vizual o soluţie posibilă, eliminând aparent orice legătură cu editarea de cod sursă. În acest mod cursantul nu se mai împiedică în particularităţile limbajului folosit ci îşi distribuie întreaga atenţie pe componenta logică în determinarea soluţiei problemei, axându-se doar pe dezvoltarea algoritmului cerut.

Prin urmare, în cadrul lecţiilor din aplicaţia noastră, utilizatorul foloseşte - iniţial fără a conştientiza - elemente care sunt greu abordabile după modelul de predare clasic, modelează algoritmi după logica proprie, în mod liber, neîngrădit de rigorile unui limbaj de programare, dar în acelaşi timp, prin acţiunile sale, "redactează" cod sursă executabil. El poate apoi, prin adnotări grafice, să scrie codul propriu, corespunzător fiecărei acţiuni şi să-l supună unei evaluări automate din cadrul MDL.

În selectarea şi generarea lecţiilor se ţine cont nu doar de nivelul de cunoştinţe ci şi de nivelul de înţelegere al fiecărui utilator. Lecţiile pot fi particularizate şi în funcţie de categoria de vârstă.

Din punctul de vedere al începătorilor, procesul de învăţământ are două componente (Figura 1):

- spaţiul de modelare, în care va primi cunoştinţele teoretice necesare şi va modela soluţii pentru problemele date spre rezolvare;
- interpretorul de comenzi.

Spaţiul de modelare este "sala de clasă" în care elevul îşi va desfăşura activitatea, scena în care el utilizează tehnici vizuale şi interactive pentru a construi forma metaforică a algoritmului cerut. După această etapă, în funcţie de şabloanele predefinite de către profesor, interpretorul identifică relaţiile dintre acţiunile descrise şi instrucţiunile corespunzătoare. Interpretorul transformă apoi acţiunile în instrucţiuni pe baza cărora se va scrie codul sursă.

Utilizatorii avansaţi au posibilitatea să interacţioneze direct cu interpretorul, descriind acţiuni prin pseudocod, sărind astfel etapa de modelare vizuală a soluţiei şi apropiindu-se de modul de lucru dintr-un limbaj de programare (Figura 2).

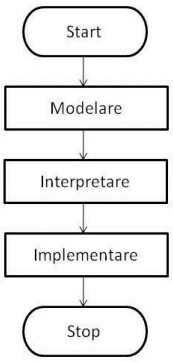

Figura 1. Diagrama procesului de învăţământ din punctul de vedere al cursantului începător

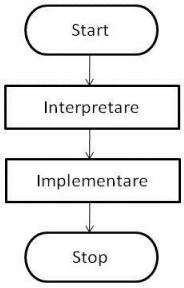

Figura 2. Diagrama procesului de învăţământ din punctul de vedere al cursantului avansat

Roluri

În vederea realizării scopului propus, sistemul va interacţiona cu trei actori, rolurile în MDL fiind administrator, cadru didactic (profesor, dascăl) şi cursant (student, elev) (Figura 3).

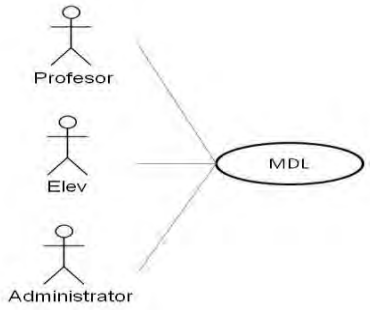

Figura 3. Roluri în MDL

Administratorul asigură integritatea resurselor şi supraveghează procesul didactic din punct de vedere tehnic, modul de autentificare şi nivelul de acces al

utilizatorilor la nucleul aplicaţiei, întreţine bazele de date etc. Administratorul aplicaţiei este cel ce adaugă unelte noi la sugestia cadrelor didactice, creează noi spaţii de modelare şi tehnologii de interacţiune între utilizator şi sistem.

Cadrul didactic (profesor sau dascăl) nu dispare din procesul de învăţământ, rolul acestuia fiind cel mai semnificativ. El dispune de documentele şcolare pe baza cărora va funcţiona întreaga aplicaţie, introduce lecţii şi urmăreşte sincronizarea lor cu programa şcolară şi planificările calendaristice actuale.

Profesorul utilizează uneltele MDL cu care va gestiona întreg procesul instructiv-educativ: alcătuieşte setul de lecţii în conformitate cu documentele şcolare curente, stabileşte noţiunile teoretice şi aplicaţiile practice care vor fi abordate în cadrul fiecărei lecţii în parte.

În generarea problemelor, dascălul crează obiecte, specifică acţiunile posibile asociate, descrie forme metaforice şi identifică legătura dintre acţiunile utilizatorilor şi instrucţiuni. Astfel, în realizarea contextului aplicativ al lecţiei, dascălul crează forme metaforice, de la simplu la complex, reprezentate printr-o gamă largă de obiecte predefinite sau create cu ajutorul uneltelor puse la dispoziţia sa de către MDL (Figura 4).

Profesorul defineşte de asemenea datele de test utilizate în validarea soluţiei şi şabloanele privind:

- relaţionarea între pseudoinstrucţiuni şi evenimentele descrise de elev;
- specificarea structurilor de date corespunzătoare metaforelor utilizate;
- constantele din forma metaforică ce vor fi înlocuite cu variabile.

Pseudocodul rezultat este convertit în cod sursă tot pe baza unor şabloane predefinite de către dascăl.

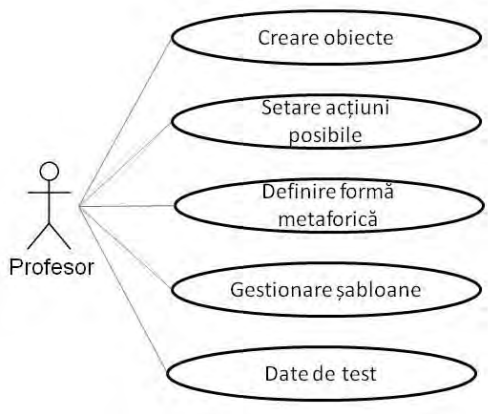

Figura 4. Rolul profesorului

Utilizatorul ţintă, cursantul (elev sau student), participă la cursuri corespunzător nivelului de instruire şi traiectoriei şcolare selectate în prealabil la înregistrarea în sistemul virtual de învăţământ.

În cadrul unei lecţii, cursantul studiază noţiunile teoretice corespunzătoare şi rezolvă probleme, fie interacţionând prin modelare vizuală, fie - în cazul celor mai avansaţi - prin descrierea soluţiei în pseudocod. În primul caz, elevul vizualizează o scenă în care are la dispoziţie o serie de obiecte predefinite de către dascăl, caracterizate prin atribute specifice şi acţiuni permise, cu care interacţionează şi pe care le utilizează în vederea modelării unei soluţii în rezolvarea problemei (Figura 5).

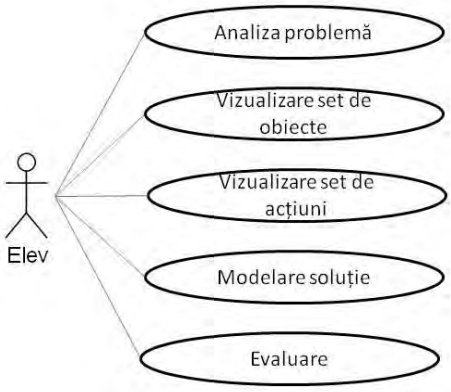

Figura 5. Rolul elevului

Caracteristicile procesului didactic:

- Uşurinţa în descrierea algoritmului

Prin modelarea vizuală a soluţiei, gândirea începătorilor nu este îngrădită de lipsa de experienţă în utilizarea unui limbaj de programare. Pe lângă această tehnică de interacţiune elev - mediu deja cunoscută şi aplicată în familiarizarea începătorilor cu domeniul informaticii, strategia didactică propusă se remarcă prin procesul de evaluare şi instruirea bazată pe analiza feedback-ului.

- Libertatea în alegerea soluţiei

Elevul nu este obligat să utilizeze o anumită metodă de rezolvare predefinită, nu există un şablon care să impună succesiunea stărilor cuplului obiect - acţiune. Dacă în final, forma metaforică descrisă de el corespunde cu cea aşteptată, atunci obţine nota maximă.

Deşi pot exista numeroase trasee alternative în modelarea unei soluţii posibile, există o singură formă metaforică finală, astfel încât procesul de evaluare este infailibil. Indiferent de drumul ales, destinaţia e aceeaşi. Fie, de exemplu, problema clasică de sortare crescătoare a elementelor unui vector unidimiensional. Se pot aplica numeroase metode de sortare, dar datele de intrare şi de ieşire rămân aceleaşi: vectorul dat respectiv vectorul sortat. Indiferent de algoritmul implementat, cât timp nu se urmăreşte eficienţa - ceea ce în cazul începătorilor nu are sens - ci doar corectitudinea soluţiei, procesul de evaluare nu va eşua.

- Tratarea erorilor de judecată

Deoarece prin modelarea vizuală a algoritmilor am eliminat posibilitatea apariţiei erorilor de sintaxă, răspunsurile eronate se datorează exclusiv erorilor de judecată. Pentru identificarea lor, se caută în setul de soluţii cea mai asemănătoare din punct de vedere al efectului acţiunilor descrise şi se face o comparaţie, pas cu pas, între stările descrise de elev şi corespondentul lor în soluţie. Se indică astfel căi alternative în traseul logic urmat în modelarea algoritmului cerut.

- Identificarea soluţiilor false

De regulă metaforele utilizate de către dascăl descriu un caz particular al unei probleme generale. Prin interacțiunea sa cu spațiul de modelare a soluțiilor, elevul poate obține un rezultat corect, o formă metaforică identică cu cea predefinită, printr-o serie de acțiuni ce descriu un caz particular și al căror efect nu poate fi generalizat.

De exemplu, sortarea în ordine crescătoare a elementelor unui vector se poate descrie prin următoarele metafore: se dă un număr constant de butoaie, iar fiecare conține o anumită cantitate de pulbere (setul de obiecte). Butoaiele pot fi interschimbate între ele oricare două câte două (setul de acțiuni). Presupunem că în urma modelării realizate de către elev se obține o aliniere a butoaielor astfel încât orice butoi din șir conține cel puțin la fel de multă pulbere ca vecinul din stânga și cel mult la fel de multă pulbere ca vecinul din dreapta. Această soluție poate fi validă, acțiunile elevului descriind o metodă de sortare general valabilă sau una falsă, aranjarea butoaielor realizându-se aleator, iar descrierea procesului de sortare fiind imposibil de generalizat. În acest caz se va proceda la ghidarea elevului spre o soluție corectă utilizând programarea prin exemple.

Identificarea acestor situații are loc în momentul în care se face trecerea la cod executabil când, în urma testării algoritmului pe baza setului de date de intrare, nu se obțin datele de ieșire așteptate în toate cazurile.

- Îmbogăţirea bazei de date

Fiecare descriere validată a algoritmului ce nu există în biblioteca sistemului va fi adaugată la setul de soluţii corespunzător problemei date. Setul de soluţii se foloseşte în procesul de feedback, proces în care elevul este ghidat, în funcţie de caz, spre cel mai apropiat răspuns corect sau spre răspunsul optim.

- Eficienţa şi optimizarea soluţiilor

În cazul în care elevul obţine o soluţie validă, sistemul afişează eficienţa algoritmului corespunzător raportată la cea maximă şi, unde e cazul, realizează o comparaţie între soluţia elevului şi soluţia optimă. Astfel elevul este îndrumat spre urmărirea performanţei în programare.

Lecţia

Didactica informaticii se orientează spre rezolvarea de probleme, utilizând prin componenta vizual-interactivă metode activ-participative, insistând pe etapa de analiză a problemei. Lecţia are două componente:

- teoretică în care are loc transmiterea de noi cunoştinţe;
- practică, bazată pe rezolvarea de probleme.

Lecţia este generată în funcţie de traseul şcolar al cursantului, de cunoştinţele şi noţiunile dobândite anterior şi nu în ultimul rând de programa analitică aflată în vigoare. În desfăşurarea lecţiilor se urmăresc o serie de activităţi de învăţare [10] precum:

- utilizarea instrumentelor informatice în modelarea unor activităţi cotidiene;
- obţinerea, în funcţie de scopul propus, a unor prelucrări complexe prin combinarea unor operaţii elementare (paşi);
- descrierea unui algoritm prin forme metaforice cât mai naturale;
- descrierea în detaliu a etapelor rezolvării problemelor din punctul de vedere al modelării algoritmilor;
- compararea diverşilor algoritmi utilizaţi în rezolvarea unei probleme pentru a sublinia eficienţa, avantajele şi dezavantajele fiecărei soluţii în parte;
- modelarea unor probleme întâlnite în viaţa de zi cu zi sau în studiul altor discipline (aplicaţii interdisciplinare);
- familiarizarea cu un limbaj de programare prin vizualizarea corespondenţei acţiunilor intreprinse cu instrucţiuni, paşii unui algoritm şi structurile de control specifice;
- proiectarea, modelarea şi implementarea unui algoritm;
- utilizarea feedback-ului în depanarea programelor;
- studierea comportamentului programelor în funcţie de diferite date de intrare.

În cadrul transmiterii de cunoştiinţe noi, se foloseşte îndeosebi programarea prin exemple sau demonstraţia. Se expun conceptele fundamentale şi se exemplifică pe un număr redus de obiecte. Apoi, prin generalizare, se trasează forma abstractă a algoritmului.

Evaluarea se realizează în etapa a doua a lecţiei, cea de rezolvare a problemelor, în fiecare din momentele sale:

- descrierea formei metaforice;
- descrierea procesului;
- descrierea pseudocodului;
- codul sursă final;
- rezultatul final, executabil.

În cazul începătorilor ce aplică în rezolvarea problemei exclusiv modelarea vizuală a soluţiei, evaluarea răspunsului se va axa pe compararea sa cu o formă metaforică predefinită de dascăl ce constituie fie soluţia unică, fie o soluţie posibilă. Procesul de evaluare include analiza acţiunilor utilizatorului şi a rezultatului returnat în comparaţie cu cel aşteptat.

Procesul didactic

Modulul de dezvoltare a lecţiilor asistate de calculator funcţionează ca un element de legătură între principalii actori ai sistemului de învăţământ, dascălul şi cursantul, şi furnizează contextul educaţional necesar pentru transmiterea respectiv însuşirea principiilor programării structurate.

În rezolvarea problemelor, în special în cazul începătorilor, se pune accentul pe modul de gândire al elevului. Se urmăreşte formarea unei abordări algoritmice, dezvoltarea capacităţii de analiză a unei problemă date, astfel încât să poată extrage din context datele de intrare şi de ieşire, să compună şi mai ales să modeleze o soluţie după propriile idei, descriind procesul printr-o formă

metaforică, fără a se preocupa iniţial cu redactarea de cod într-un limbaj de programare. Acţiunile intreprinse vor descrie instrucţiuni de pseudocod ce vor fi apoi transcrise, pe baza unor şabloane predefinite de către profesor, în cod sursă executabil.

Dupa însuşirea noţiunilor teoretice, trecând la partea practică a lecţiei, elevul studiază problema ce urmează a fi rezolvată, vizualizează setul de obiecte disponibile şi construieşte prin acţiunile sale o formă metaforică. Sistemul alcătuieşte procesul descris de acţiunile utilizatorului şi compară răspunsul obţinut cu soluţiile existente în baza de date.

În cazul în care răspunsul este corect iar traseul specificat de cursant în modelarea soluţiei nu există în baza de date, se va reţine soluţia dată, contribuind la creşterea complexităţii analizatorului de soluţii. Asociind fiecare acţiune cu instrucţiuni corespunzătoare, procesul descris de utilizator este apoi transpus în pseudocod şi mai departe, corelat cu şabloanele de cod sursă existente rezultă codul sursă executabil.

Dacă soluţia finală a utilizatorului nu este corectă, atunci se indică erorile logice apărute şi se afişează indicii pentru obţinerea modelului aşteptat.

Feedback-ul este un element exenţial în sistemul de învăţământ. Pentru fiecare cursant se crează şi se actualizează după fiecare lecţie un set de caracteristici evidenţiate în timpul procesului didactic precum capacitatea de asimilare şi aplicare în practică a noilor cunoştinţe, viteza de lucru în modelarea soluţiei, tipuri de erori întâlnite şi frecvenţa lor.

STUDIU DE CAZ

Plan de lecţie

Unitatea de învăţare: Descrierea algoritmilor.

Conţinuturi: Structura alternativă (de decizie).

Competenţe specifice: Analiza problemei. Identificarea datelor de intrare / ieşire, stabilirea paşilor de rezolvare a problemei. Modelarea şi reprezentarea algoritmilor. Principiile programării structurate în elaborarea algoritmilor.

Activităţi de învăţare: Modul de execuţie al instrucţiunii alternative. Exemple de utilizare din viaţa cotidiană. Urmărirea modului de execuţie a instrucţiunii pas cu pas. Aplicaţii: determinarea minimului şi maximului a două numere, verificarea parităţii unui număr, rezolvarea ecuaţiilor de gradul I şi II, divizibilitate etc.

Lecţia propriu-zisă

Consideraţii teoretice: definirea şi descrierea structurii alternative.

Aplicaţia practică: selectarea maximului a două numere.

Forma metaforică: compararea a două cantităţi şi selectarea celei mai grele. Umplerea a două găleţi cu apă, compararea lor cu ajutorul unui cântar şi selectarea celei mai grele.

Setul de obiecte: robinet, apă, găleţi, cântar.

Acţiuni posibile: deschidere/închidere robinet, umplere găleată cu apă, aşezare găleată pe cântar, selecţie găleată.

Desfăşurarea lecţiei

Studentul modelează soluţia problemei folosind obiectele disponibile şi acţiunile permise în cadrul lecţiei. El poate să obţină forme metaforice acceptate ca soluţie mergând pe trasee alternative, ducând totuşi spre rezultatul aşteptat. În acest caz se va completa baza de date a sistemului cu un nou algoritm.

În cazul în care utilizatorul nu reuşeşte sa ajungă la forma finală, are posibilitatea să vizualizeze forme metaforice intermediare pentru a ieşi din impas şi de asemenea poate urmări o trasare pas cu pas a modelării soluţiei corecte.

Bazându-se pe demersul determinist de rezolvare a problemei date, MDL realizează asocierea obiectelor şi acţiunilor din procesul de modelare vizuală cu variabilele şi instrucţiunile unui limbaj de programare. Lecţia prezentată anterior se interpretează conform Tabelului 1.

Tabelul 1. Determinarea maximului a două numere date

Modelarea vizuală a soluţiei prin acţiuni	**Instrucţiunile echivalente în pseudocod**	**Cod sursa C++**
vizualizarea obiectelor disponibile (două găleţi şi un cântar) şi acţiunile permise	două variabile şi o structură alternativă	float x,y;
umplerea găleţilor cu apă	citirea valorilor a două variabile	cin>>x>>y;
cântărirea găleţilor	test logic if-then-else	if (x<y) cout<<x; else cout<<y;
selectarea găleţii mai grele	afişarea soluţiei	

Din acest exemplu simplu reiese faptul că utilizatorul controlează aparent modelarea soluţiei într-un mod vizual, interactiv, fără să aibă vreo legatură cu editarea codului sursă. Controlul este aparent deoarece tocmai aici apare marea problemă în programare, în special în cazul utilizatorilor începători – se pierd în soluţii. Pentru a evita afundarea în mulţimea de posibilităţi de rezolvare şi implicit prevenirea pe cât posibil a erorilor logice, utilizatorul este constrâns în a utiliza doar anumite obiecte date.

Am adoptat acest demers determinist pentru a conduce programatorul începător spre găsirea unei soluţii corecte prin limitarea posibilităţilor şi implicit a erorilor de

judecată. Rolul profesorului devine cu atât mai important cu cât el este responsabil de a încărca în baza de date - corespunzător fiecărei probleme propuse - obiecte şi acţiuni ce permit modelarea cât mai multor soluţii, anticipând şi totodată ghidând elevul spre modelarea soluţiei optime.

GENERALIZARE

Analog exemplului descris anterior poate fi modelată orice lecţie de programare, principiul urmărit fiind general valabil: utilizatorul scrie codul sursă prin acţiuni şi nu prin instrucţiuni. Cunoscând relaţia acţiune-instrucţiune, MDL generează pseudocod, iar apoi, pe baza unor şabloane existente în baza de date, furnizează cod sursă executabil. La următoarea trasere, codul este vizualizat de utilizator, fiecărei acţiuni fiindu-i adnotată o instrucţiune.

Această abordare se poate extinde la orice lecţie de programarea calculatoarelor sau în studiul unor tehnici de programare. Pot fi explicate concepte din programare precum tipuri de date, structuri de control, tipuri structurate, pointeri, arbori, grafuri, liste, subprograme, se pot studia tehnici de programare, proprietăți etc. De exemplu, următoarea formă metaforică: utilizatorul crează cercuri de diverse dimensiuni prin modelare grafică, ilustrând diverse proprietăți precum instanţierea unui obiect dintr-o clasă (clasa cerc), duplicarea, încapsularea, moştenirea, moştenirea multiplă etc.

Pentru avansaţi, din moment ce utilizatorul s-a familiarizat deja cu manevrarea obiectelor, nivelul de detaliu creşte, lecţia conţine mai multe obiecte văzute acum ca variabilele echivalente într-un mediu de programare şi continuă să modeleze, după preferinţă în mod grafic sau pseudocod, dar pe o cale crescând nedeterministă. MDL va conţine în continuare unul sau mai multe soluţii şi va analiza rezultatul obtinut de secvenţa de acţiuni a utilizatorului. Se vor obţine astfel soluţii noi, care vor fi memorate şi folosite în lecţiile următoare. Deoarece în această abordare solutia e vazută ca o anumită stare a obiectelor date/utilizate, se poate testa orice soluţie propusă de utilizator. În cazul în care cursantul nu găseşte o soluţie, el poate accesa soluţii posibile sau îndrumări parţiale prin adnotări grafice pas cu pas.

CONCLUZII

Abordarea prezentată în această lucrare se vrea a fi un pas mai aproape de traversarea barierei dintre utilizator şi programator faţă de metodele existente la ora actuală, în special la noi în ţară, în didactica informaticii. Ne-am axat pe aspectul introductiv, pe familiarizarea studenţilor neinstruiţi cu programarea calculatoarelor, pe componentele care realizează interfaţa dintre aceşti utilizatorii şi mediul de programare. Aplicaţia prezintă unelte uşor de utilizat de către cadrele didactice în realizarea lecţiilor şi un motor de evaluare automată a soluţiilor cursanţilor. Aceştia beneficiază de un sistem de învăţare uşor accesibil, care le stimulează dezvoltarea gândirii algoritmice şi îi fereşte de constrângerile unui limbaj de programare. Totodată se realizează asocierea între modelarea vizuală şi redactarea de cod, obţinându-se în final cod sursă executabil.

Extinderi ale modelului propus

Demersul determinist, potrivit începătorilor, trebuie înlocuit cu unul tinzând spre nedeterminism o dată cu evoluţia cursantului, deoarece utilizatorul expert nu mai are nevoie să modeleze vizual algoritmul, ci are abilităţile necesare pentru a scrie codul sursă propriu direct într-un mediu de programare.

Automatizarea proceselor. Trecerea de la o problema particulară la un algoritm general, de pildă, pentru studiul de caz prezentat în lucrare, generalizarea la aflarea maximului a n numere date; sortarea a n elemente; parcurgerea unui graf etc

Modul de interacţiune. Extinderea dispozitivelor utilizate în realizarea mediului educaţional, de la scene 3D vizualizate pe monitor, interacţiune prin tastatura, mouse şi/sau tabletă grafica, la un întreg mediu educaţional virtual proiectat 3D şi controlat prin mănuşă.

Includerea lecţiilor generate de nucleul MDL într-o platformă de eLearning facilitând învăţământul la distanţă dar menţinând, chiar prin construcţia lor, controlul asupra feedback-ului şi a evoluţiei liniare, în acelaşi timp facile, a capacităţii de dezvoltare a algoritmilor de către utilizatori de pretutindeni.

REFERINŢE

1. The Lifelong Kindergarten group, http://llk.media.mit.edu/index.php [2010]
2. MIT Media Lab, http://www.media.mit.edu/ [2010]
3. The Intel Computer Clubhouse Network, http://www.computerclubhouse.org/ [2010]
4. Lego Mindstorms NXT. http://mindstorms.lego.com/en-us/default.aspx [2010]
5. PicoCricket computers, http://www.picocricket.com [2010].
6. Jonathan S. Harbour, Microsoft Visual Basic Game Programming for Teens, 2nd Ed, ISBN-13: 978-1598633900, Course Technology PTR, 2007.
7. Adobe Authorware 7. http://www.adobe.com/products/authorware/ [2010]
8. Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Chris Scaffidi, Mary Shaw, Susan Wiedenbeck. "The State of the Art in End-User Software Engineering", ACM Computing Surveys. Accepted for publication. http://faculty.washington.edu/ajko/papers/Ko2010EndUserSoftwareEngineering.pdf [2010]
9. M Grötschel, W R Pulleyblank, Mathematics of Operations Research, Volume 11, Issue 4, ISSN:0364-765X, INFORMS, Linthicum, Maryland, USA, 1986, pp. 537 – 569.
10. Programa şcolară pentru Informatică intensiv, clasa a IX-a. http://www.edu.ro/index.php/articles/curriculum/c556+559+580+/ [2010]