Милан Вугделија

ЈЕДАН ПРЕДЛОГ ПРОМЕНА У НАСТАВИ ПРОГРАМИРАЊА У СРЕДЊИМ ШКОЛАМА

Програмирање има своје место у средњошколским наставним програмима, пре свега природног смера. Могуће је да ће се оно ускоро појавити и као изборни предмет у основним школама. Велика и несумњива корист која се стиче учењем програмирања јесте развијање логичког мишљења, боље разумевање појма модела (нарочито кроз објектно оријентисано програмирање), развијање способности апстракције и апстрактног мишљења и друге менталне способности. Са друге стране, кроз учење програмирања у школи може се стећи важна и чврста основа за савладавање програмирања као професије. Ипак, програмирање је у школама тако конципирано да се много више ради на остваривању првог од ова два циља. Наша пажња ће овде такође бити усмерена ка остваривању тог првог циља, посвећеног развоју.

Идеја да се програмирање учи у школама је у пракси наишла на извесне, можда неочекивано велике проблеме. Ђаци често с муком стигну да савладају наредбе улаза и излаза, `if` наредбу и једну наредбу циклуса (обично је то `for` циклус, као најједноставнији). И таман када треба да се вежбају у састављању неких једноставнијих алгоритама и њиховом прецизном исказивању, за то више нема времена. Не треба ни говорити о атмосфери која у таквој ситуацији влада, о незадовољству и наставника и ученика. С правом се може поставити питање: зашто се програмирање учи тако тешко и споро?

Програмирање само по себи није тако једноставно. Формализам који треба савладати представља озбиљан проблем, поготово за некога ко се са њим први пут среће. Списак свега онога што је у програмирању тешко тиме сигурно није завршен. Али, запитајмо се шта се поред разних објективних разлога оваквом стању, појављује као неповољна околност, која би се можда могла и уклонити? Шта можемо да учинимо да се учење програмирања олакша ученицима?

Свакоме ко нешто учи, изузетно је важно да види примере употребе знања које стиче. То му даје мотив да истраје у савладавању предмета који изучава. Свако има потребу да сагледа бар неку сврху стицања знања којим се бави. Није битно да то буде једина, па чак ни најважнија сврха (гледано стратешки). Битно је да се смисленом и занимљивом применом стеченог знања подстакне мотив и воља за учењем.

У настави програмирања каква је данас, задаци са бројевима и рачунањем су далеко најчешћи: примера ради, ту су прости бројеви и са њима у вези прости

делиоци датог броја, савршени и пријатељски бројеви, затим Питагорине тројке бројева, аритметичка средина, Фибоначијев низ, верижни разломци и многи слични задаци. О смислу и могућим применама свих тих формула и бројева ђацима се врло мало говори, а чак и кад би им се говорило више, то и даље није употреба рачунара какву они познају. Да и не помињемо да се у настави програмирања користи углавном оперативни систем DOS, који се данас вероватно не користи ни за шта друго. Све то изазива одбојност ученика према предмету и ствара тешкоће у настави. Многи данашњи ученици су се врло рано срели са рачунаром (чак у предшколском узрасту), а од самог почетка користе искључиво графичке оперативне системе. Примена рачунара која је блиска искуству ученика је углавном рекреативног карактера: играње игара, слушање музике, гледање филмова и сл. Већ са дизајном било које врсте и обрадом слике искуства су мања, а неким интензивним рачунањем се ретко ко бавио. Стога, ако желимо да осавременимо наставу програмирања, да је прилагодимо узрасту и интересовањима ученика, да је повежемо са њиховим искуством и праксом, морамо укључити савременије аспекте програмирања.

После програмирања у паскалу (а то је програмски језик који је највише изучаван у школама) најлогичнији наставак је објектни паскал и окружење Delfi. У овом окружењу врло једноставно се креирају модерни програми, који користе графички кориснички интерфејс (GUI), цртање, бит-мапе, репродукцију звука и приказивање готових анимација у стандардним форматима, приказивање графикона, рад са registry базом, .INI фајловима, базама података, као и многе друге атрактивне могућности.

Прилично је јасно да би оваквим начином рада многи дидактички принципи наставе могли бити квалитетније и потпуније испоштовани. Илуструјмо ово на примеру принципа очигледности у настави: приликом предавања наредби циклуса можемо употребити процедуре за цртање кругова или правих линија (које су ученици упознали раније) уз одређено успоравање програма и постићи да ученици дословно виде како наредба циклуса ради. Погодни примери су цртање мете, шаховске табле и сл. Исто важи и за друге дидактичке принципе, као што су већ поменути принцип прилагођености узрасту (више графике и мултимедије уопште, а мање нумерике), принцип свесне активности ученика у настави, принцип повезаности теорије и праксе и други.

Може изгледати да су неки принципи, попут *систематичности и поступности у настави* оваквим начином рада угрожени. И овај принцип се у највећој мери може испоштовати, са једне стране постепеним усложњавањем примера и постепеним изношењем све сложенијих детаља о некој теми, а са друге решавањем најпре најједноставнијих, а затим сложенијих алгоритама.

Са до сада изложеним разматрањем би се у принципу сложили и многи наставници. Међутим, није сасвим јасно када, како и колико треба укључити Delfi (или неко друго слично окружење) у наставу програмирања. Прецизније, у вези са тим проблемом се поставља неколико питања:

- Да ли је боље прво учити програмирање у турбо паскалу па онда прећи на Delfi, или од почетка учити Delfi?

- Да ли је могуће са постојећим фондом часова обухватити све што је потребно за Delfi програмирање?
- Да ли се раније дефинисан циљ, посвећен развоју личности, одвајањем потребног броја часова на изучавање окружења и интерфејса, запоставља на рачун много мање вредног, претежно чињеничног знања о датом окружењу?

Наиме, јасно је да је креирање корисничког интерфејса у програмима углавном једноставно, поготово са алгоритамске стране, али да упознавање истог захтева извесно време, пре свега због обимности градива. Стога није једноставно помирити противречности које се овде јављају: не потрошити превише времена, а изложити материјал који је кудикамо обимнији од оног до сада предаваног, и поврх тога наћи и довољно времена за бављење алгоритмима.

Ево неких смерница, чијим би се праћењем ово у доброј мери могло испунити.

♢ *Направити избор онога што ће бити презентирано ученицима.* Ко год је, макар и површно, погледао палете компоненти или својстава и догађаје разних објеката у објект инспектору (испитивач објеката), разуме да изучавање програмирања у Delfi окружењу може однети много више времена него што било који средњошколски програм може да обезбеди. Не треба ни помињати многобројне техничке детаље и теоријске концепте, који долазе са сложенијим оперативним системом и објектно оријентисаним програмирањем. Зато потреба за избором није спорна. Ради остваривања циља који разматрамо није ни битна комплетност излагања, већ да се окружење упозна толико да у њему може да се ради. Избор могу да донекле учине комисије које се баве овим послом и писци уџбеника прилагођених настави у средњој (евентуално основној) школи. Коначан избор ипак треба да буде право и слобода сваког наставника, према његовој процени услова у којима ради, интересовања ученика и др.

♢ *Не излагати сваки део градива једнако детаљно.* Први сусрет са визуелним и другим компонентама захтева нешто више времена за навикавање на нове појмове и рад са објект инспектором. Како разне компоненте имају многа заједничка својства и друге сличности, касније се може само поменути у пар речи оно што је карактеристично за неку нову компоненту. Када се ученици навикну на појмове компонента, контрола, својство, метода, догађај и сл. и када се увежбају у употреби објект инспектора, система помоћи и штампане литературе, моћи ће чак и самостално или у групама да откривају оно што им је потребно да би могли да користе неку компоненту. Овакав приступ може бити користан не само због рационалније употребљеног времена, већ и зато што подстиче самосталност, сналажљивост, уочавање аналогија и друге способности код ученика. Такав приступ подржава ученике да се баве и даљим изучавањем проблематике у складу са својим интересовањима. Врло често је ученицима довољан сасвим мали подстицај, да самостално открију нове интересантне и корисне могућности и израде и презентирају примере или чак врло сложене програме. Са друге стране, ако ученици не показују интересовање, ни детаљније излагање неће бити од велике користи.

♢ *Особености окружења излагати упоредо са општим основама програмираа.* Погодним повезивањем тема и избором примера може се постићи да се више различитих ствари научи истовремено. На пример, приликом упознавања

са `if` наредбом ученици могу да науче да користе опционо дугме (CheckBox), упоредо са `case` наредбом радио групу, обичну и комбиновану листу (ListBox, ComboBox), са низовима мемо поље и детаљније класу TStrings, која се појављује у многим компонентама, при раду са фајловима дијалог за отварање и чување фајла и сл. Цртање се може вежбати од самог почетка, кроз алгоритме линијске структуре, али и са наредбама гранања и циклуса. Исто тако, у оквиру једног примера може се илустровати рад више различитих компоненти. Употреба компоненти на елементарном нивоу је толико једноставна да се без проблема може разумети и научити успут, не одвајајући сувише времена за њихово изучавање.

Последња смерница сугерише *рад у Delfi окружењу од самог почетка*. Поред уштеде времена услед (временског) преклапања неких садржаја, постоји још много аргумената за овакав став:

♢ Примери и садржаји би били примеренији и самим тим интересантнији ученицима, о чему је већ било речи. Нумерика и обрада текста у стринговима и фајловима је данас само мали сегмент примене рачунара. Визуелна блискост са оним што раде (графичко окружење), препознавање сврхе и осећај моћи да се за кратко време направи привлачан програм су разлози због којих се може очекивати значајан скок у мотивисаности ученика и њиховом задовољству у раду. Ово је већ примећено и потврђено у досадашњој, релативно скромној пракси.

♢ Основе алгоритамског решавања проблема се могу стицати и кроз примере у којима се ради и са графиком, сликама, звуком, анимацијом и разним ефектима, а не само са бројевима (и понекад словима). На тај начин се не мора изгубити ништа од онога сто представља тежиште учења програмирања, а то је пре свега развој одређених менталних способности кроз бављење алгоритмима.

♢ Након савладавања чак интелектуално тежег градива (смишљање и записивање алгоритама различите сложености), уводни примери рада са компонентама и графиком делују сувише једноставно и празно. Другим речима, у таквом редоследу изучавања, ово градиво на неки начин „виси“, јер стиже са закашњењем. Нема никаквог смисла да се тако једноставне и атрактивне ствари уче пред крај бављења програмирањем. *Ово знање није тешко усвојити због некакве компликованости, него због масе углавном једноставних техничких детаља и чињеница којима треба овладати. Такав проблем не захтева зрелост ученика да прихвати градиво, него време да се слични садржаји, који су једноставни али обимни, међусобно раздвоје у времену, као и да се основе „слегну“ пре евентуалног бављења напреднијим темама.*

♢ Објектно оријентисано програмирање (ООР) постоји већ пар деценија и данас је без конкуренције најприхваћенији концепт програмирања. Бавити се данас програмирањем, а не упознати ООР значи одрећи се контакта са стварношћу. Мада има и врло сложених разматрања у вези са ООР, неки елементи су веома једноставни. Одлагање упознавања основних појмова и концепата ООР и навикавања на њих може знатно отежати улазак у ову област.

На овом месту желим да изнесем и нека лична искуства, која су ме у првом тренутку изненадила, али која су у ствари лако објашњива. Када сам са ђацима користио Turbo Paskal, догађало се да они током годину и по дана чежњиво

траже Delfi, односно Windows програмирање. За све то време сматрали су градиво које уче потпуно застарелим и зато сувишним, неки од њих чак и бесмисленим. Сличних примедби је било и у вези са задацима. Када смо коначно ушли у Delfi, настало је својеврсно разочарање открићем да је сочан плод који су дуго ишчекивали више личио на сјајну, али празну љуску. Схватили су да је све оно што им је „пунило очи“, у суштини врло једноставно постићи, а да је и даље потребно на практично исти начин испрограмирати понашање компоненти и програма у целини. У извесном смислу обрнут случај је био са ђацима који од почетка користе Delfi окружење. Ја сам оставио њима на вољу да задатке решавају пишући конзолне или Windows програме. После кратког времена већи број ученика је самостално прешао на писање конзолних програма! Једноставно, након што су савладали основе графичког корисничког интерфејса (GUI), нису имали жеље и потребе да се и даље њиме интензивно баве, већ су желели да се посвете суштини конкретног проблема. Важно је, међутим, то што су ученици схватили праву улогу и једноставност креирања графичког корисничког интерфејса, да би се ослободили неких заблуда. Мој утисак је да када препознају GUI као „шарени папир“ за умотавање поступака које треба смислити и записати, ученици се са више воље баве алгоритмима као суштином програмирања.

Мој је утисак да када сагледају програмирање из ове перспективе, ученици се лакше могу придобити за дискусију и усмено решавање неких врло тешких проблема. Наиме, сматрам да се не мора инсистирати на томе да се свака идеја реализује до краја и да се по њој обавезно напише програм који ради. Напротив, и тек како има смисла да се понекад поставе и такви задаци, за које већи број ђака не може да напише програм, али може да смисли, објасни и ручно изведе поступак на мањем примеру. Чини ми се да је све то лакше остварити када ученици имају јаснију представу о томе колико је то далеко (или близу) од онога што они свакодневно виђају или онога што могу да реализују до краја.

У том смислу, потврђује се (мада статистички недовољно за извођење финијих закључака) да Delfi окружење ђацима не представља проблем, већ им даје мотив за бављење темама које су ближе суштини и пружа им могућност да дају јаснији смисао својим активностима. При томе се треба чувати опасности да се оде у крајност и да се настава сувише посвети упознавању окружења. Добром организацијом може се (и треба!) постићи да остане времена и за бављење алгоритамски тежим задацима. Наравно, пожељно је такве задатке интерпретирати и поставити на атрактивнији начин, на пример као програмирање логичких игара.

Ниједна промена у било којој активности није сасвим једноставна, па ни прелазак на Delfi. Он захтева труд и обуку наставника, као и већу ангажованост у припреми часова. Посебно је важна улога наставника у одмеравању и процењивању колико може и треба да се поствети Delfi окружењу у конкретним условима. Међутим, прелазак на Delfi се до сада показао као напор који се и те како исплати, имајући у виду пре свега скок у мотивисаности и заинтересованости ученика, а тиме и у резултатима рада.