

Методичні основи створення педагогічних програмних засобів.

Масове оснащення шкіл комп'ютерами почалося після появи інформатики в навчальних планах середньої школи. На початковому етапі ЕОМ використовувались в основному для навчання програмуванню. За останні роки зміст курсу інформатики потерпів значних змін.

Майже одночасно з появою в школі комп'ютерів були створені програми, призначені для навчання школярів програмуванню. Потім з'явилися програми для навчання іншим предметам. За такими програмами та інструментальними засобами закріпився термін „педагогічні програмні засоби”.

Педагогічні програмні засоби (ППЗ) – це сукупність комп'ютерних програм, призначених для досягнення конкретної цілі навчання.[1] ППЗ є головною частиною комп'ютерного програмно-методичного комплексу, який включає в себе, крім ППЗ, методичний та дидактичний супровід даних програм.

Сучасна шкільна освіта будується на класно-урочній системі, яка мало змінилася з часів Яна Каменського. За час існування системи виробилися способи організації фронтальної роботи, ефективні як при викладанні навчального матеріалу, так і при контролі засвоєних знань. В той же час і саме засвоєння навчального матеріалу і швидкість його строго індивідуальні і потребують індивідуального підходу, особливо для вироблення навичок. Особливо цей етап навчання і викликає найбільші складності в учнів та вчителів. Тому, якщо використання ОТ може якимсь чином послабити напругу, виникаючу на даному етапі, то це буде безумовно виправдано.

Оскільки комп'ютер створювався як засіб управління технічними та іншими системами, він майже відразу знайшов застосування в управлінні якістю навчання.

В області застосування засобів ОТ в навчальному процесі можна виділити два підходи. Згідно першого поява такого потужного засобу обробки інформації має вести до перегляду змісту і форм освіти. Другий полягає в тому, що комп'ютер – це просто технічний засіб навчання і тому не може визначати ні змісту, ні форм освіти, проте це не означає, що він не може впливати на них.

Комп'ютер як відомо являється засобом обробки інформації. Його робота основана на чисельному кодуванні інформації і використанні математичних методів її опрацювання. Йому неважливо, якого роду ця інформація, головне щоб вона була формалізована. В той же час сучасні технічні і програмні засоби дозволяють подавати інформацію в різних формах: текстовій, графічній (діаграми, графіки, малюнки і т.п.), звуковій і навіть в механічній. І що дуже важливо, швидкодія комп'ютера дозволяє

йому вести опрацювання інформації в режимі реального часу – оперативно відгукуючись на дії користувача або показники приладів. Таким чином, для використання комп'ютера в різних галузях головним є формалізація інформації і способів її опрацювання, що і є однією з перших проблем з якою доводиться стикнутися розробникам ППЗ. Все інше – це технічні проблеми, які можуть бути вирішені більш менш задовільно.

Проте основною проблемою в використанні ППЗ в навчанні є те, що програмні засоби спочатку створюються, а потім розпочинаються способи знайти їм місце в навчальному процесі. І тепер, оскільки існує попит, розробляються програмні продукти для підтримки викладання різних предметів. Проте ці програми створюють частіше за все не так, як це потрібно учню та вчителю, а так, як це зручно і зрозуміло розробнику. Так наприклад, чи не першими ППЗ стали численні „автоматизовані навчаючі системи” – просто тому, що подібні програмні засоби використовувалися на підприємствах і в момент появи комп'ютерів в масовій школі стали актуальними задачі автоматизації та створення „автоматизованих робочих місць”. Зручність і простота розробки викликали появу великої кількості електронних задачників - текстів з викладеним навчальним матеріалом, які супроводжувалися в кращому випадку графічними ілюстраціями; багато контролюючих програм, в яких зазвичай потрібно з декількох запропонованих відповідей вибрати вірну.

В зв'язку з цим можна вказати на системи, інколи дуже потужні, призначені для розв'язання навчальних задач. Основний їх недолік в тому, що в таких системах незрозуміло, навіщо взагалі потрібен учень – майже все за нього робить програма.

Можна з впевненістю казати, що лише невелика частина ППЗ можуть використовуватися при викладанні в школі.

В якості ще однієї з причин існуючого положення можна назвати наступне. Як показує вивчення існуючих ППЗ, вони створюються або вчителями, або програмістами мало зв'язаними з школою. В першому випадку програми виявляються безпорадними з точки зору програмування, використовують часткові можливості комп'ютера, проте найчастіше в їх основі покладена грамотна методика і програми бувають дидактично обґрунтовані. В іншому випадку програми можуть бути створені бездоганно з точки зору техніки програмування, проте є безпорадними з дидактичного боку.

Разом з тим методика викладання кожного навчального предмету в свою чергу враховує особливості відповідної науки, тому правомірно говорити про методичні вимоги до ППЗ, які передбачають специфіку конкретної науки і відповідного їй навчального предмету. Визначаючи педагогічні вимоги, які ставляться перед ППЗ, необхідно враховувати також особливості вибору тематики ППЗ, аргументоване певними методичними цілями, і забезпечувати перевірку педагогічної ефективності використання ППЗ.

Крім цього при розробці ППЗ необхідно враховувати ще й ряд інших факторів: вікові та індивідуальні особливості учнів, забезпечення

доброзичливої і тактовної форми звернення до учня, можливість повторних звернень до програми в випадку невдалої спроби. Все це обумовлює позитивний фон спілкування користувача з ЕОМ, визначаючи ергономічні вимоги до змісту і оформленню ППЗ.

Отже можемо бачити, що розробка ППЗ представляє собою дуже складний процес, який потребує колективної праці не тільки вчителів, методистів, програмістів, але і психологів, гігієністів, дизайнерів.

Виходячи з цього можна перерахувати основні вимоги, які ставляться до ППЗ: [2]

- педагогічні вимоги (дидактичні, методичні, обґрунтування вибору тематики, перевірка на педагогічну цілісність використання і ефективність застосування);
- технічні вимоги;
- ергономічні вимоги;
- фізіологічно-гігієнічні вимоги;
- естетичні вимоги;
- вимоги до оформлення документації.

Проблема створення і використання комп'ютерних навчальних програм залишається актуальною. Педагогічна цінність і якість ППЗ залежить від того, наскільки повно враховується при його розробці комплекс вимог, які ставляться перед ними. Як було зазначено вище, створення високоякісних ППЗ - колективна робота, яка потребує великих зусиль спеціалістів різних областей.

В даний час немає єдиної класифікації ППЗ, незважаючи на те що в багатьох працях в залежності від методичних цілей, реалізація яких виправдує введення ППЗ, виділяють серед них наступні типи:

1. інформаційно-довідкові ППЗ, за допомогою яких подається інформація про явища і процеси, що вивчаються;
2. навчально-наставницькі ППЗ, які призначені для керівництва пізнавальною діяльністю учнів. Також не виключається можливість подачі додаткової, нової, інформації;
3. імітаційно-моделюючі ППЗ, які фактично демонструють діючі моделі явищ і процесів, що вивчаються;
4. ігрові ППЗ з повним набором інформації для представлення і пояснення змісту гри та аналізу отриманих результатів;
5. контролюючі ППЗ, які призначені для організації процесів повторення раніше вивченого матеріалу, контролю якості його засвоєння, застосування у практичній діяльності;
6. програми-тренажери (тренувальні програми);
7. програми для проблемного навчання;

Проте найчастіше зустрічаються комбіновані ППЗ, які включають в собі декілька зазначених вище типів.

Питанням застосування НІТ в тому числі і ППЗ у навчанні математики в школі присвячено роботи А.П.Єршова, В.М.Монахова, А.М.Довгялло,

В.Г.Болтянського, М.І.Жалдака, М.Х.Розова, С.А.Ракова, А.В.Пенькова, Ю.В.Горошка, О.Б.Жильцова, В.В.Дровозюк, Т.О.Олійник, Є.М.Смирнової, М.С.Голованя, Т.В.Зайцевої; і у вузі – В.П.Дяконова, В.І.Клочка, Ю.С.Рамського, Ю.В.Триуса, Ю.Г.Лотюка та інших.

Питаннями відбору та конструювання педагогічних програмних засобів та створенням інших дидактичних засобів на основі НІТ займалися М.І.Жалдак, Т.А.Сергеева, Є.В.Чубров, Ю.В.Горошко, А.В.Пеньков, С.А.Раков, Т.О.Олійник, Н.О.Калініна, О.В.Вітюк та ін.

Проте для багатьох шкільних дисциплін питання адаптації інформаційних технологій до програми навчання та пошуку найкращих форм організації навчання в умовах їх застосування залишається невирішеним, чим і пояснюється той факт, що в школі, крім інформатики, комп'ютери використовуються дуже мало. Це стосується як математики, так і інших шкільних предметів. Здебільшого у вищенаведених дослідженнях вивчаються можливості застосування ІТ до окремих навчальних тем, деяких аспектів їх впливу на учбову діяльність учнів, їхній розумовий розвиток, пізнавальний інтерес тощо. Цілісної системи впровадження ІТ у навчальний процес ще немає. Між тим стає зрозумілим, що на даному етапі інформатизації освіти не можна повністю відкинути традиційні форми навчання, потрібно шукати “розумний компроміс” між традиційними формами та ІТ.

Зважаючи на вищесказане постає задача навчити майбутніх вчителів, зокрема інформатики і математики, створювати ППЗ для підтримки викладання свого, а можливо і інших предметів або хоча б ставити адекватні вимоги до програмістів. Оскільки при створенні такого програмного засобу розробник має не лише володіти предметом, в галузі якого відбувається розробка та вміння програмувати, йому необхідно розумітися на науково методичних основах, які будуть використовуватися при проектуванні і створенні ППЗ.

Наприклад розглянемо деякі особливості, покладені при створенні ППЗ Gran2D, які можуть дати загальне уявлення і розуміння аналізу проблем, що мають вирішуватися програмним засобом. Даний програмний засіб розроблений на мові програмування Object Pascal і тому подальший виклад буде базуватися на об'єктно-орієнтованому підході, покладеному в цій мові програмування. Розгляд будемо проводити на дещо спрощеній формі ПЗ, з невеликою кількістю можливостей. Проте побудуємо каркас нашої програми в подальшому який можна бути доповнений новими можливостями.

Оскільки в нас буде стояти необхідність візуалізації геометричних побудов то нам необхідно визначити клас, який буде містити характеристики нашої робочої області. Таким клас буде TGrPlace, який буде базуватися на стандартному класі TImage.

```
TGrPlace=class(TImage)
Private
    FPutOsi: Boolean;
    FDrawGrid: Boolean;
```

```

FYNameOsi: String;
FXNameOsi: String;
FColorOsi: TColor;
FColorObl: TColor;
Fxc: Integer;
Fyc: Integer;
FRolX: Integer;
FRolY: Integer;
FZoom: Extended;
FAZoom: Integer;
FTecDX, FTecDY : Extended;
AB : Array[1..5] Of Record
                ex, ey : Extended;
                End;
FlagDrawScene : Boolean;
LastX, LastY : Double;
Public
constructor Create(AOwner: TComponent); override;
procedure ScreenToAbsZ(x, y: Integer; var xx, yy: Extended);
procedure AbsToScreenZ(x, y: Extended; var xx, yy:Extended);
    overload;
procedure AbsToScreenZ(x, y: Extended; var xx, yy:Integer);
    overload;
procedure SetZoom(n : Extended);
procedure DrawLine(x1,y1,x2,y2:Extended; Lt : Byte=0);
procedure Draw;
procedure GrMouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure GrMouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure GrMouseUp(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure Save(f : TIniFile; Sec : String);
procedure Load(f : TIniFile; Sec : String);
property ColorObl : TColor read FColorObl write FColorObl;
property PutOsi : Boolean read FPutOsi write FPutOsi;
property ColorOsi : TColor read FColorOsi write FColorOsi;
property DrawGrid : Boolean read FDrawGrid write FDrawGrid;
property XNameOsi : String read FXNameOsi write FXNameOsi;
property YNameOsi : String read FYNameOsi write FYNameOsi;
property TecDX : Extended read FTecDX;
property TecDY : Extended read FTecDY;
property AZoom : Integer read FAZoom;
End;

```

Даний клас як робоча область, яка буде відобразити прямокутну систему координат (ПДСК) буде характеризуватися: кольором робочої області - ColorObl; необхідністю відображення осей координат - PutOsi, координатної сітки - DrawGrid, та їх кольорами - ColorOsi, підписами назв осей - XNameOsi та YNameOsi. Деякі з характеристик є лише для читання, оскільки їх значення залежать від деяких параметрів і не можуть змінюватися за межами даного класу. Так наприклад AZoom - кількістю точок в 1 см., яка

залежить від параметрів екрану; TecDX та TecDY координати точки в ПДСК над якою знаходиться курсор миші. Частина полів для роботи класу є прихованими, розміщені в області private, і призначені лише для внутрішнього функціонування класу, проте змінити деякі з них можна за рахунок виклику відповідних методів: FZoom - коефіцієнтом збільшення, масштаб, метод зміни SetZoom.

Крім методу зміни масштабу в створюваний клас було включено методи переведення координат з ПДСК в координати робочої області та навпаки ScreenToAbsZ, AbsToScreenZ; метод побудови лінії DrawLine за її типом (пряма – Lt=0, промінь – Lt=1, відрізок – Lt=2) та перебудови системи координат Draw; методи по опрацюванню подій миші, та методи для збереження в файл і завантаження параметрів графічної області.

Йдучи далі можна зазначити, що будь який програмний засіб має працювати з певною групою об'єктів: математичні функції ($y(x)$, $g(x,y)$, $r(f)$, ...), геометричні об'єкти (точка, лінія, коло). Виходячи з таких потреб необхідно визначити деякі загальні особливості, характеристики всіх цих об'єктів і можливо визначити деякий головний об'єкт (батьківський клас), який би містив ці особливості, а інші об'єкти (класи) будуть базуватися на головному, беручи від нього основні характеристики і будуть доповнюватися власними специфічними лише для нього характеристиками. Зрозуміло що таких батьківських класів програмний засіб може містити декілька.

Так наприклад при створенні ППЗ Gran2D було визначено батьківський клас TParentObj, від якого будуть походити всі інші класи геометричних об'єктів.

Опис даного класу має наступний вигляд

```
TParentObj=class(TObject)
private
    FName : String;
    FVisible : Boolean;
    FTypeObj: ShapeType;
    FReCalc: Boolean;
    FSelect: Boolean;
protected
    function GetType: String; virtual; abstract;
public
    RefCount : Integer;
    Ref : Array[1..2] Of Integer;
    GrPlace : TGrPlace;
    Constructor Create;
    procedure ShowSett(M : TStrings; Move : Boolean=False);
        virtual; abstract;
    function ShowInfo : String; virtual; abstract;
    procedure Draw; virtual; abstract;
    procedure Save(f : TIniFile; Sec : String); virtual;
    procedure Load(f : TIniFile; Sec: String; List : TStrings);
        virtual;
    property TypeObj : ShapeType read FTypeObj write FTypeObj;
```

```

property Name : String read FName write FName;
property Visible : Boolean read FVisible write FVisible;
property Select : Boolean read FSelect write FSelect;
property ReCalc : Boolean read FReCalc write FReCalc;
End;

```

Для даного класу були виділені такі специфічні особливості: будь який об'єкт має мати ім'я Name, приналежність до певного типу TypeObj (точка, пряма, відрізок, промінь, та ін.), показчик чи є об'єкт в даний момент видимим Visible, та виділенням Selected, та показчик необхідності його перерахунку ReCalc (якщо зміниться положення об'єктів від яких він залежить). Оскільки наші об'єкти будуть міститися в деякому списку, то для ідентифікації того, від яких об'єктів залежить даний використовуємо масив їх позицій в списку Ref та кількість таких об'єктів RefCount. Та поле, як посилання, на об'єкт графічної області, для його відображення.

При визначенні методів для батьківського класу необхідно передбачити можливість роботи з файлами, збереження та завантаження об'єктів, методи Save та Load. Крім того визначені методи, які будуть виводити коротку ShowInfo та повну ShowSett інформацію про об'єкт. Та метод Draw призначений для відображення об'єкту. Деякі з методів є прихованими, зокрема GetTypeNames, який повертає текстовий запис типу об'єкту і використовується лише при його збереженні.

Як можна бачити деякі методи є абстрактними, оскільки не можуть бути описані для загального класу і в подальшому будуть описані в класах нащадках.

Отже після того, як буде описаний батьківський клас з визначеними загальними особливостями, можна приступати до опису похідних класів, класів нащадків, які будуть характеризувати або сам об'єкт або знову бути деяким проміжним класом для визначення і доповнення деяких особливостей для групи інших об'єктів.

Так наприклад при визначенні класу точка він може бути доповнений власними особливостями серед яких можна виділити:

- координати точки в ПДСК – DPx, DPy та на площині відображення – DGx, DGu (оскільки буде необхідність знати як явні координати точки так і ті відносно яких вона буде зображатися на робочій області), зважаючи на це необхідно буде визначити процедуру такого перерахунку, яку розмістимо в методах запису значень координат площини відображення;
- візуальні властивості такі як розмір (радіус) - DSize точки, колір зовнішньої лінії - DColor, колір – DFill та необхідність зафарбування - DFillShow;
- поле яке буде відповідати за підпис точки DLab : TTextPoint (визначення класу TTextPoint буде показано пізніше);
- параметри підпису: необхідність відображення підпису – DnameShow та координат - DCoorShow.

Та зрозуміло необхідно буде підмінити (довизначити) методи описані в батьківському класі та при необхідності визначити власні.

Крім того передбачимо при створенні об'єкту точка можливість передачі її координат для спрощення створення.

```
TPointObj=class(TParentObj)
private
  FDGx, FDGy : Double;
  FDCoorShow: Boolean;
  FDNameShow: Boolean;
  FDFillShow: Boolean;
  FDSIZE: Byte;
  FDFill: TColor;
  FDColor: TColor;
  procedure SetDGx(const Value: Double);
  procedure SetDGy(const Value: Double);
  procedure SetVisible(const Value: Boolean);
  function GetTypeNames: String; override;
  procedure ShowCaption;
public
  DPx, DPy : Integer;
  DLab : TTextPoint;
  constructor Create(TypeDot : ShapeType; APlace :
    TGrPlace=nil); overload;
  constructor Create(x1, y1 : Double; TypeDot : ShapeType;
    APlace : TGrPlace=nil); overload;
  Destructor Destroy; override;
  function ShowInfo : String; override;
  procedure ShowSett(M : TStrings; Move : Boolean=False);
    override;
  procedure Draw; override;
  procedure Save(f : TIniFile; Sec : String); override;
  procedure Load(f : TIniFile; Sec: String; List : TStrings);
    override;
  property DGx : Double read FDGx write SetDGx;
  property DGy : Double read FDGy write SetDGy;
  property DColor : TColor read FDColor write FDColor;
  property DSize : Byte read FDSIZE write FDSIZE;
  property DFill : TColor read FDFill write FDFill;
  property DFillShow : Boolean read FDFillShow write
    FDFillShow;
  property DNameShow : Boolean read FDNameShow write
    FDNameShow;
  property DCoorShow : Boolean read FDCoorShow write
    FDCoorShow;
End;
```

Аналогічно клас прямої може бути до визначеним до такого виду:

```
TLineObj=class(TParentObj)
private
  Ax, By, Cz : Extended;
  FName1: String;
  FName2: String;
```



```

FLSize: Byte;
FLStyle: Byte;
FLColor: TColor;
function GetTypeNames: String; override;
public
Gx1, Gy1, Gx2, Gy2 : Extended; //координати точок на прямих
Px1, Py1, Px2, Py2 : Integer; //координати точок на прямих
constructor Create(TypeLine : ShapeType; APlace :
TGrPlace=nil);
function ShowInfo : String; override;
procedure ShowSett(M : TStrings; Move : Boolean=False);
override;
procedure SetPropLine2Dot(xx1, yy1, xx2, yy2 : Extended);
procedure Draw; override;
procedure SetNameDot(s1,s2 : String);
procedure Save(f : TIniFile; Sec : String); override;
procedure Load(f : TIniFile; Sec: String; List : TStrings);
override;
property LStyle : Byte read FLStyle write FLStyle;
property LColor : TColor read FLColor write FLColor;
property LSize : Byte read FLSize write FLSize;
End;

```

Серед специфічних властивостей визначені:

- тип лінії LStyle (пряма, промінь, відрізок);
- коефіцієнти рівняння отримуваної прямої Ax, By, Cz;
- координати точок на яких будується пряма в ПДСК Gx1, Gy1, Gx2, Gy2; та точок робочої області Px1, Py1, Px2, Py2;
- FName1, FName2 – назви точок на яких базується лінія та метод їх внесення, оскільки поля є захищеними.

Крім того виділено метод SetPropLine2Dot за допомогою якого визначаються коефіцієнти прямої.

Крім батьківського класу для всіх геометричних об'єктів в ППЗ Gran2D був визначений батьківський клас для текстових написів, які можуть міститися в робочій області програми. Серед таких текстових об'єктів в програмі можуть зустрічаються: підпис точки, текстовий напис та ін.

```

TTextObj=class(TLabel)
private
XOld, YOld : Integer;
FlagMove : Boolean;
procedure LabelMouseUp(Sender: TObject; Button:
TMouseButton; Shift: TShiftState; X, Y: Integer);virtual;
procedure LabelMouseDown(Sender: TObject; Button:
TMouseButton; Shift: TShiftState; X, Y: Integer);virtual;
procedure LabelMouseMove(Sender: TObject; Shift:
TShiftState; X,Y: Integer); virtual;
public
GrPlace : TGrPlace;

```

```

constructor Create(AOwner: TComponent); overload; override;
constructor Create(AOwner: TComponent; APlace : TGrPlace);
overload;
procedure SetPopupMenu(APopupMenu : TPopupMenu);
procedure Save(f : TIniFile; Sec : String); virtual;
abstract;
procedure Load(f : TIniFile; Sec: String; List : TStrings);
virtual; abstract;
end;

```

Даний клас характеризуються такими загальними особливостями напису для всіх своїх нащадків як характеристикою приналежності до графічної області GrPlace, прихованими полями XOld, YOld та FlagMove для можливості переміщення напису по робочій області.

Серед визначених методів можна виділити метод SetPopupMenu – прикріплення власного контекстного меню до об'єкту; методи по роботі з файлами Load, Save, які є абстрактними та приховані методи обробки подій миші.

Як нащадки від TTextObj визначимо клас для підпису точки TTextPoint та клас для текстового напису TTextGraph, перевизначивши деякі методи та ввівши додаткові поля.

```

TTextPoint=class(TTextObj)
public
constructor Create(AOwner: TComponent); overload; override;
procedure GraphCaption(const Value: String; AColor : TColor;
ASize : Integer; AName : String);
end;

TTextGraph=class(TTextObj)
private
FTexts: TStrings;
procedure SetTexts(const Value: TStrings);
public
constructor Create(AOwner: TComponent); overload; override;
Destructor Destroy; override;
procedure Save(f : TIniFile; Sec : String); override;
procedure Load(f : TIniFile; Sec: String; List : TStrings);
override;
property Texts : TStrings read FTexts write SetTexts;
end;

```

Але зрозуміло не завжди можна визначити деякий батьківський клас, оскільки програма може містити об'єкти, які не будуть мати нащадків так наприклад в повній версії Gran2D серед одиничних можна виділити об'єкти по роботі з динамічними виразами, кнопками, макроконструкціями.

В подальшому можна буде доповнювати вже визначені класи новими можливостями, або на їх основі створювати нові класи для збільшення кількості робочих об'єктів.

Отже можна бачити, що на першому етапі створення програмного засобу необхідно проаналізувати той сегмент завдань обраної предметної

галузі, які ми будемо намагатися реалізувати за його допомогою. Визначившись з потребами програмного засобу сформувавши перелік необхідних об'єктів та їх взаємозв'язок між собою. Окреслити ті специфічні характеристики, якими будуть володіти об'єкти та методи які ці об'єкти зможуть виконувати.

Література

1. Волинський В.П., Козлакова Г.О. Методичні рекомендації до використання педагогічних програмних засобів у навчальному процесі. – К.: НПУ ім. М. П. Драгоманова, 2007. – 59 с.
2. Вострокнутов И.Е. Оценка визуальных сред на экране монитора или почему болят глаза при работе на компьютере. // Информатика и образование. - 2002. - №-1.- С. 64 - 67.