

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Співаковський О.В., Лещинський О.Л.

НАВЧАЛЬНА ПРОГРАМА
“СВІТ ЛІНІЙНОЇ АЛГЕБРИ”

Херсон – 2001

Співаковський О.В., Лещинський О.Л. Навчальна програма «Світ лінійної алгебри». — Херсон: Вид-во Херсонського педагогічного університету, 2001. — 42 с.

В даному посібнику на прикладі навчальної програми «Світ лінійної алгебри» описано ідеологію підготовки сучасних навчальних програм.

Рекомендовано аспірантам та викладачам вищих навчальних закладів.

Рецензенти:

Усенко Віталій Михайлович, доктор фізико-математичних наук,
професор;

Клочко Віталій Іванович, доктор педагогічних наук, професор.

*Рекомендовано до друку Вченою радою
Херсонського педагогічного університету*

ВСТУП

Найкращий момент почати статтю наступає тоді, коли ви її успішно закінчили. В цей час вам стає зрозумілим, що конкретно ви хотіли сказати.

Марк Твен

Даний посібник присвячено проблемі автоматизації навчання (зокрема навчання елементам лінійної алгебри). Сам по собі термін «автоматизація» настільки поширений, що навряд чи потребує додаткових пояснень. Зрозуміло, що автоматизація навчання – це передача комп'ютеру деяких функцій, які раніше виконувались викладачем. До їх числа можна віднести і ті, які стали відносно простими, звичними, які не вимагають спеціальних інтелектуальних зусиль. З одного боку, навчаючі програми можуть застосовуватись тільки для розрахунків в громіздських задачах (калькулятивна мета). З іншого боку, вони можуть дійсно на деяких етапах замінити або серйозно допомогти викладачу навчити студента конкретному елементу навчання. Можна стверджувати, що автоматизації практично піддаються всі обчислювальні операції і навчаючі модулі. Необхідні тільки педагогічно-професійні постановка задачі і її реалізація.

Взагалі кажучи, автоматизація педагогічного процесу вимагає нових підходів в самій методиці викладання конкретного предмету. Наприклад, при автоматизованому навчанні будь-яка неточність на початкових стадіях може виявитись непоміченою за багаточисельними формальними побудовами траєкторії навчання і автоматично приведе до невірних результатів, хоча останні будуть сприйматися, як ті, які заслуговують на довіру. Тому одна з основних задач – це розробка методів об'єктивного виділення цілісних педагогічних систем як необхідної умови, яка гарантує позитивні результати автоматизованого навчання і в подальшому ефективного управління цими результатами. В зв'язку з цим саме поняття цілісності навчання потребує спеціального розгляду, тому що має практичне і методологічне значення. По-перше при вивченні педагогічної системи кожний раз виникає необхідність якось виділити і обмежити

об'єкт навчання, причому мати в основі цих операцій достатньо міцну теоретичну базу, а не тільки характеристики, які змінюються з часом. По-друге, це поняття конкретизує, тобто робить більш розгорнутим наше уявлення про педагогічну систему, яка вивчається. По-третє, ефективність керуючих заходів прямо пов'язана з рівнем цілісності об'єкту управління. Тому помилка в виділенні границь і структури системи, як такого об'єкту, може привести до значних додаткових затрат ресурсів і часу, і навіть поставити під загрозу саму можливість досягнення мети системи. Тому, будь-яка навчаюча програма, взагалі кажучи, повинна мати вхідний тест (фільтр) можливостей студента, який вказує на можливості навчання за допомогою даної програми. Цей тест – достатньо серйозна проблема, яка потребує окремих досліджень і авторами розуміється, але не розглядається. Навчаючі програми і системи повинні розроблятися найбільш кваліфікованими спеціалістами (викладачами) і програмістами, тому що необхідні знання і досвід, а також інтуїція для точного формулювання проблеми, задачі знаходження критеріїв. Н.Вінер (1983 р.) попереджав, що ЕОМ корисна рівно настільки, наскільки важливі і корисні ідеї її користувачів. Тому викладач і програміст формулюють задачу навчання і розробляють план її розв'язання, а комп'ютер використовується на початковій стадії навчання для наступної перевірки правильності, уточнення і реалізації ідей авторів. Очевидно, що існує і зворотній зв'язок: досконалість тактичних засобів дозволяє змінювати стратегію навчання, ставити перед методикою нові задачі, суттєво змінювати зміст старих. На сьогодні значна кількість педагогічних задач (задач навчання конкретному предмету зокрема) в тій чи іншій мірі автоматизована. Існує достатньо навчаючих систем, тестів, електронних книг, з лінійної алгебри зокрема. Тому постає питання, чи потрібне оновлення цього програмного забезпечення?

По-перше, існують різні підходи і погляди на зміст навчання лінійній алгебрі в педагогічному вузів зокрема. А тому програмне забезпечення для цих підходів повинно бути різним.

По-друге, існують різні погляди на системи вправ для педагогічних вузів зокрема.

По-третє, існують різні педагогічні (методичні) системи, кожна з яких, взагалі кажучи, має право на існування.

Тому автори вважають, що в методичний комплекс лекторів і викладачів, що ведуть практичні заняття, на сучасному рівні повинно входити комп'ютерне методичне забезпечення, тобто навчаючі, контролюючі програмні модулі або цілі системи взагалі.

Основні педагогічні задачі автоматизації процесу навчання можна розглядати в наступній послідовності:

- виділення основних об'єктів матеріалу, який вивчається і визначення їх співвідношення;
- опис внутрішньої будови вказаного навчального об'єкту (теоретичного матеріалу, підбраного набору вправ, розробленого спеціально призначеного тесту і т.п.);
- виявлення причинної обумовленості тієї чи іншої будови навчального об'єкту;
- встановлення функцій навчальних об'єктів і законів їх функціонування;
- оцінка ресурсів навчальних об'єктів і педагогічних можливостей їх використання на різних етапах навчального процесу;
- прогноз варіативності навчальних об'єктів в процесі їх природної еволюції і педагогічного використання;
- створення автоматизованих систем управління навчальними об'єктами або моделювання навчальних об'єктів з заданими властивостями.

Майже всі ці задачі на сьогодні вже більш чи менш традиційні для педагогічної науки і так чи інакше нею розв'язуються. Але, навіть сьогодні, постає природне питання про необхідність автоматизації навчального процесу взагалі. По-перше, немає чіткої відповіді на це питання психологів, психіатрів, педагогів-теоретиків та інших спеціалістів, що мають відношення до процесу навчання. По-друге, існують суттєві труднощі на цьому шляху, серед яких технічні, фінансові,

кадрові, психологічні і т.п. Які ж аргументи на користь автоматизованих педагогічних технологій? Їх теж достатньо багато. Серед них:

- необхідність якісного зрушення в коректності обумовленості і адекватності розв'язання традиційних педагогічних і методологічних задач; виникає і підсилюється ця необхідність під впливом конструїзації педагогічної науки, її зростаючої орієнтації на пріоритетний розв'язок найважливіших проблем національної освіти;
- суттєво зростає відповідальність методики за правильність обумовлення рекомендацій, які нею пропонуються, навіть незначні помилки чи малопомітні недоліки загрожують легко придбати риси глобальних проблем, на розв'язання яких будуть потрібні величезні засоби і значний час.

Звідси впливають зрозумілі вимоги до конструктивного навчального процесу — максимально повне і об'єктивне описання об'єкту вивчення, що охоплює всю сукупність фактів, які відносяться до питання, що розглядається. Всі факти, параметри, зв'язки, що впливають на педагогічний об'єкт повинні бути враховані, описані і оцінені. Але задовольнити ці вимоги, взагалі кажучи, дуже складно в силу величезної кількості інформації, яку необхідно зібрати, систематизувати, усвідомити та використати. Причому тут виникає цікавий ефект. Розмір адекватного інформаційного масиву практично не залежить від просторових масштабів об'єктів, що вивчаються.

У традиційному уявленні організація навчального процесу та професійного навчання студентів — це ієрархічна структура. Нажаль, формування такої структури відбувається тривалий період і тому за рахунок консервативності не встигає враховувати можливості більш швидкої системи розробки та впровадження програм. І тому, на сьогоднішній день, ми маємо з кожного фаху набір дисциплін, які не завжди зв'язують організацію і структуру з часом і не корелюють між собою. Нажаль, дуже часто немає чітко окреслених горизонтальних зв'язків між змістовною частиною цих дисциплін та автоматизацією процесу навчання. Формування навчаючих систем і комплексів

здійснюється, як правило, стихійно. Немає загального методичного підходу, відсутні єдині системні позиції при розробці таких систем на кафедрі зокрема і в окремому ВУЗі взагалі. На сьогодні для успішної роботи викладачів та керуючих органів ВУЗів в сфері автоматизації навчального процесу доцільно розв'язати задачу створення єдиного інтегрованого середовища з базового, прикладного та навчаючого програмного забезпечення. Це задача складна, але її треба починати розв'язувати. Широке розповсюдження і використання ПЕОМ надало новий поштовх подальшому розвитку автоматизованих навчальних систем (АНС) та навчаючих комплексів (АНК). Під АНС розуміють людинно-машинну систему для організації і управління пізнавальної діяльності в процесі навчання. До АНС входить технічне, програмне, інформаційне і методичне забезпечення. Як технічне забезпечення використовується ПЕОМ, об'єднані в локальну сітку. Програмне забезпечення АНС призначене для підготовки навчаючих програм, підтримки роботи в локальній сітці й управління базою даних. Інформаційне забезпечення АНС включає навчаючі програми і протоколи навчання, які містять диференційовану і інтегровану інформацію про роботу студентів з навчаючими програмами і її наслідки. Методичне забезпечення АНС включає повні алгоритми складання сценаріїв навчаючих програм відповідно вибраній системі технологій, створення цих навчаючих програм і управління процесом навчання. АНК поєднує в єдиний логічний простір відповідні АНС.

ПОГЛЯД НА СТВОРЕННЯ ПОДІБНИХ ПРОГРАММ

Вибір середовища

При виборі середовища ми вирішили не враховувати системне і прикладне програмне забезпечення платформи Mac (Macintosh). Через високу вартість комп'ютерів даного типу вони не одержали настільки широкого поширення в мережі навчальних закладів.

Вибір операційної системи

Найбільш популярними операційними системами (ОС) на сьогоднішній день є Unix (Linux, Santa Cruz Operation і ін.) і Windows (Windows 98, Windows NT, Windows 2000, Windows XP і більш пізні).

Основна частка використання операційних систем типу Unix припадає на сервери, хоча Linux останнім часом усе більше знаходить популярність і як станційна ОС. Недостатня популярність цієї ОС пояснюється недоліком прикладного програмного забезпечення під неї.

Пізні версії ОС Windows (Windows 98 і вище) широко поширені в мережі навчальних закладів, та й на домашніх комп'ютерах воліють ставити саме Windows. Виробник ОС даного типу (фірма Microsoft) не зазнає серйозних фінансових потрясінь, багато «третіх» фірм працює над створенням і удосконалюванням прикладного ПО саме під Windows. Застосування конкретної версії залежить від обмежень на обсяг оперативної пам'яті, розміру вільного дискового простору на твердому диску та інших характеристик персонального комп'ютера. Отже, ми зупиняємося на операційній системі Windows.

Вибір середовища програмування

При розгляді варіативності середовища програмування ми будемо враховувати засоби розробки Windows-додатків, виходячи з обраної

операційної системи. На сьогоднішній день найбільш популярними в даній іпостасі є Visual Basic, Visual C++, Delphi.

Дані засоби розробки – це продукти одного класу, забезпечені в цілому достатнім набором засобів і компонент для створення розвинutih Windows-додатків. Кожен програмний продукт має свої відмінні риси.

Можна говорити, що Visual Basic прекрасно інтегрований з рідним для Windows Microsoft Office і він порівняно простий для використання. Але його продуктивність і сервісні можливості для програмування поступаються можливостям двох інших мов.

Мова C++ широко розповсюджена. Ця мова являє собою могутній засіб для розробки найрізноманітніших програм. Більшість сучасних операційних систем (Unix, DOS, Windows) написані саме на цій мові. Програми, написані на C++ поступаються по продуктивності тільки програмам, створеним мовою Асемблера. C++ є об'єктно-орієнтованим розширенням C, і завдяки підтримці об'єктного програмування перетворився в найбільш могутній засіб для розробки найрізноманітніших програм. Але ця мова занадто складна для вивчення і використання. Безліч нюансів, що відслідковуються в інших мовах автоматично, повинен контролювати сам програміст. Крім того, багато типів даних, що максимально просто реалізуються в Паскалі чи Бейсику, у C++ реалізовані порівняно незручно. У першу чергу це відноситься до рядків. Рядки в C++ представляються не у виді окремого типу (як в інших мовах), а як масиви знаків. Звідси і складності з будь-якими операціями над ними. У C++ відсутні засоби для перевірки границь масиву, і там, де в Паскалі можна одержати попередження компілятора, у C++ нічого не відбудеться доти, поки розроблювач (у випадку помилки) замість необхідних даних не одержить невідоме «сміття». Тому налагодження програм забирає значний час.

Delphi надає користувачу зручні засоби для написання додатків під ОС Windows. Delphi - не просто мова. Це надзвичайно могутнє і зручне інтегроване середовище розробки (IDE), що надає користувачу дружню, інтуїтивно зрозумілу, просту у використанні і разом з тим багатофункціональну оболонку. Ефективність коду, сгенерованого

компілятором Delphi не на багато поступається C++. Object Pascal, що лежить в основі Delphi, широко розповсюджений у навчальних програмах з програмування у ВУЗах. Недолік Delphi є продовженням його достоїнств. Використання зручного Паскаля утруднює взаємодія з операційною системою, тому що системною мовою є C++, а його синтаксис і, можна сказати «філософія», серйозно відрізняються від аналогічних характеристик Паскаля. Це призводить до деяких труднощів використання API (Application Program Interface – стандартний інтерфейс, що дозволяє додаткам верхнього рівня працювати з пакетом комунікаційних протоколів).

Виходячи з перерахованих вище міркувань, у першу чергу через популярність, як засіб розробки вибираємо Delphi.

Основні задачі програми:

- автоматизація знаходження рангу матриці;
- автоматизація знаходження оберненої матриці;
- автоматизація знаходження розв'язку систем лінійних алгебраїчних рівнянь матричним методом;
- автоматизоване навчання знаходження рангу матриці;
- автоматизоване навчання знаходження оберненої матриці;
- автоматизоване навчання знаходженню розв'язку систем лінійних алгебраїчних рівнянь матричним методом;
- ілюстрація написання подібних програм.

ОПИС СТРУКТУРИ ПРОГРАМИ

Опис логічної структури програми

Додаток містить комплект процедур і функцій, що умовно можна розділити на наступні функціональні блоки:

- блок візуального інтерфейсу – сукупність процедур і функцій, що забезпечують візуальне представлення даних на екрані (шрифт, колір, фон, текст);
- обчислювальний блок - сукупність процедур і функцій, що забезпечують реалізацію математичних формул;
- блок взаємодії з користувачем - сукупність процедур і функцій, що забезпечують реакцію на дії користувача (натискання кнопок, вибір елементів меню і т.д.);
- блок контролю – сукупність команд перевірки вихідних і проміжних даних на припустимі значення;
- блок повідомлень – повідомлення користувачу і «експерт».

Опис модулів програми

Додаток складається з ряду зв'язаних файлів:

Елемент	Опис	Файл	Призначення
Проект	Опис проекту	Pras.dpr	Файл проекту . Цей файл містить вихідний текст головної програми додатка. Тут створюється екземпляр головної форми додатка, а також ініціюється автоматично створювана форма FTools
		Pras.cfg	Файл конфігурації проекту Цей файл містить конфігурацію проекту і використовується компілятором при трансляції

Елемент	Опис	Файл	Призначення
		Pras.dof	Файл опцій проекту. Цей файл містить поточні установки проекту – налаштування компілятора і компоновщика, імена службових каталогів, умовні директиви і параметри командного рядка
		Pras.res	Файл ресурсів додатка. Цей файл містить ресурси додатка (піктограми, графічні зображення, курсори чи миші рядки)
Головна форма	Головна форма додатка	PW_Main.dfm	Текстове представлення форми
		PW_Main.pas	Модуль форми
		PW_Main.ddp	Файл діаграм, створюваних на сторінці діаграм вікна Редактора Коду
Форма «Робочий стіл»	Форма для введення вихідних даних і відображення кроків роботи з матрицями	PW_1.dfm	Текстове представлення форми
		PW_1.pas	Модуль форми
		PW_1.ddp	Файл діаграм, створюваних на сторінці діаграм вікна Редактора Коду
Форма для ініціювання операції над матрицями	Форма для вибору типу операції над матрицями і завдання необхідних параметрів	PTools.dfm	Текстове представлення форми
		PTools.pas	Модуль форми
		PTools.ddp	Файл діаграм, створюваних на сторінці

Елемент	Опис	Файл	Призначення
			діаграм вікна Редактора Коду
Форма для тексту підказок експерта	Форма відображення тексту підказок експерта	PExpertDialog.dfm	Текстове представлення форми
		PExpertDialog.pas	Модуль форми
		PExpertDialog.ddp	Файл діаграм, створюваних на сторінці діаграм вікна Редактора Коду
Інші модулі		PMatrClassDef.pas	Модуль з описом класу Tmatr
		Pmath.pas	Модуль з реалізацією математичних алгоритмів
		PConsts.pas	Модуль констант

Опис візуального інтерфейсу

При проектуванні візуального інтерфейсу вирішувались наступні проблеми:

- графічні елементи інтерфейсу повинні бути знайомими по роботі з Windows;
- основна робота для простоти повинна вестися на одній формі;
- форма не повинна бути перенасичена графічними елементами;
- активні графічні елементи повинні бути виділені (шрифтом, кольором);
- графічні елементи, якими в даний момент не можна користатися, повинні знаходитися в стані «не доступні» (Disabled) (альтернативним представляється рішення приховування таких елементів (зробити їх невидимими).

У програмі використовуються наступні стандартні візуальні компоненти Delphi (позначені у категоріях класів):

- TGroupBox (вікно групи) – для іменованої області форми з групою візуальних компонентів;
- TLabel (мітка) – для написів;
- TButton – для звичайних кнопок;
- TSpeedButton – для кнопок з фіксацією натиснутого стану;
- TStringGrid (таблиця рядків) – для ручного введення вихідних даних;
- TSpinEdit (вікно редагування з лічильником) – для завдання розмірності вихідної матриці;
- TRadioGroup (група радіокнопок) – для вказівки операції над матрицями;
- TEdit (вікно редагування) – для введення даних;
- TBitBtn (кнопка з графікою) – кнопки для діалогу (експерт);
- TMemo (багаторядкове вікно) – для тексту експерта.

Крім цього в додатку використовується елемент TPopupMenu (спливаюче меню), у якому продубльовані функції кнопок, а також розроблений у рамках даної програми клас TMatrix для візуалізації матриць (побудований на основі класів TGroupBox і TStringGrid).

Додаток поєднує три візуальні форми:

- головна форма – власне робочий стіл;
- форма для вибору операції над матрицями;
- форма для тексту підказок експерта.

Опис обчислювальних алгоритмів та їх взаємозв'язків

Алгоритми розв'язання задач сконцентровані в модулі PMath.pas.

Метод знаходження оберненої матриці методом Гаусса реалізується функцією Gauss, текст якої приведений нижче.

```

function
Gauss(adr:Adres;nado_e,flExpert:boolean;CurMatr,EdMatr:TMatrix):inte
ger;

        { 100 -продовжувати далі}
        { 1..6 - нульовий рядок }
        { 1000 - Gauss відробив }
        { 999 - Процедура була перервана запитом
користувача}

var
i,t,e,g,r:integer; ch, ch1, zn, zn1:longint;
Eto_Sistema, flExit:boolean;
s:string;
j1,j2:integer;
EdAdr:Adres;
RealNStl:integer;
iCol:integer;
begin
    EdAdr:=EdMatr.MatrDataAdr;
    Eto_Sistema:=CurMatr.SystemEquation;
    flExit:=false;

    with adr^ do
    begin

        RealNStl:=R_stl;
        if Eto_Sistema then
            r:=R_str
        else r:=R_stl;
        R_stl:=r;

        Result:=100;

        for i:=1 to R_str do

```

```

begin
  if Nul_Stroka(adr,i) then
  begin
    Result:=i;
    break;
  end;

```

```

If matr_ch[i][i]=0 then
Begin
  for t:=i+1 to R_str do
    if matr_ch[t][i]<>0 then
      begin
        if FW_1.flExpert then
          begin
            Row_Exchange(adr,i,t);
            CurMatr.RefreshData;
            if not FW_1.IsExpertDialogContinue(
              ExpertDialog3S1+IntToStr(i)+'+',IntToStr(e)) then
              begin
                flExit:=true;
                break;
              end
            end
          else
            Row_Exchange(adr,i,t);

            dop_ch:=-dop_ch;
          end;

          for e:=i+1 to r do
            if matr_ch[i][e]<>0 then
              begin

```



```

if flExpert then
begin
Col_Exchange(adr,i,e);
CurMatr.RefreshData;
if not FW_1.IsExpertDialogContinue(
ExpertDialog3S2+IntToStr(i)+' '+IntToStr(e)) then
begin
flExit:=true;
break;
end
end
else
Col_Exchange(adr,i,e);

dop_ch:=-dop_ch;
end;

```

```

Result:=i;
End
Else
Begin
if nado_e=false then g:=i+1 else g:=1;

for t:=g to R_str do
if (matr_ch[t][i] > 0) and (t > i) then
begin

```

```

Mult_Fractions(matr_zn[i][i],matr_ch[i][i],matr_ch[t][i],matr_zn[t][i],ch,z
n);

```

```

if EdMatr < nil then
begin
with EdMatr.MatrDataAdr^ do

```

```

Mult_Fractions(matr_zn[i][i],matr_ch[i][i],matr_ch[t][i],matr_zn[t][i],ch1,
zn1);

```

```

    end;

```

```

    if flExpert then

```

```

        begin

```

```

            if (zn=1) or (ch=0) then s:=IntToStr(ch)

```

```

            else s:=Concat(IntToStr(ch)+'/'+IntToStr(zn));

```

```

            Add_Rows(adr,i,t,-ch,zn,1);

```

```

            if EdMatr <> nil then

```

```

                with EdMatr do

```

```

                    Add_Rows(MatDataAdr,i,t,-ch,zn,1);

```

```

                CurMatr.RefreshData;

```

```

                EdMatr.RefreshData;

```

```

                if not FW_1.IsExpertDialogContinue(

```

```

                    ExpertDialog3S3+IntToStr(t)+'=

```

```

                    '+ExpertDialog3S4+' '+IntToStr(i)+

```

```

                    '* -'+s+' '+ExpertDialog3S4+' '+IntToStr(t) )

```

```

then

```

```

    begin

```

```

        flExit:=true;

```

```

        break;

```

```

    end

```

```

    end

```

```

    else

```

```

        Add_Rows(adr,i,t,-ch,zn,zn);

```

```

    end;

```

```

    if flExit then break;

```

```

End;

```

```

end;

```

```

if not flExit then

```

```

for t:=g to R_str do
  if (matr_ch[t][t]<>1) or (matr_zn[t][t]<>1) then
    begin
      ch:=matr_zn[t][t];
      zn:=matr_ch[t][t];

Mult_Fractions(matr_ch[t][t],matr_zn[t][t],ch,zn,matr_ch[t][t],matr_zn[t][
t]);
      if EdMatr<>nil then
        begin
          for iCol:=1 to r do
            with EdMatr.MatrDataAdr^ do

Mult_Fractions(matr_ch[t][iCol],matr_zn[t][iCol],ch,zn,matr_ch[t][iCol],
matr_zn[t][iCol]);
          end;
        end;
      if flExpert and not flExit then
        begin
          CurMatr.RefreshData;
          EdMatr.RefreshData;
        end;

      if not flExit then
        Result:=1000
      else
        Result:=999;

      R_stl:=RealNStl;

    end;
  end;
end;

```

Функція реалізує алгоритм побудови оберненої матриці методом Гауса за допомогою приєднаної одиничної матриці, причому враховуються тільки реальні стовпчики матриці (для розв'язання системи рівнянь стовпчик, у якому знаходяться вільні члени, участі в ітераційних процесах не бере).

Функція використовує наступні процедури/функції:

✓ процедура перестановки рядків Row_Exchange:

```
procedure Row_Exchange(adr:Adres;n1,n2:integer); {Str_Per}
var i:integer;
begin
  with adr^ do
    begin
      for i:=1 to r_stl do
        FractionExchange(matr_ch[n1,i],matr_zn[n1,i],matr_ch[n2,i],matr_zn[n2,
          i]); {obmen}
        FractionExchange(Ms1_Ch,Ms1_Zn,Ms2_Ch,Ms2_Zn); {obmen}
      end;
    end;
end;
```

✓ процедура FractionExchange виконує перестановку блоків:

```
procedure FractionExchange(var
  Numerator1,Denominator1,Numerator2,Denominator2:longint);
{Obmen}
var a:longint;
begin
  a:=Numerator1;
  Numerator1:=Numerator2;
  Numerator2:=a;
  a:=Denominator1;
  Denominator1:=Denominator2;
  Denominator2:=a;
end;
```

✓ процедура перестановки колонок Col_Exchange:

```

procedure Col_Exchange(adr:Adres;n1,n2:integer); {Stb_Per}
var i:integer;
begin
  with adr^ do
    begin
      for i:=1 to r_str do
        FractionExchange(matr_ch[i,n1],matr_zn[i,n1],matr_ch[i,n2],matr_zn[i,n
        2]); {obmen}
        FractionExchange(Mb1_Ch,Mb1_Zn,Mb2_Ch,Mb2_Zn); {obmen}
        i:=neiz[N1_Stb];neiz[N1_Stb]:=neiz[N2_Stb];neiz[N2_Stb]:=i;
      end;
    end;
end;

```

✓ процедура множення дробів Mult_Fractions

```

procedure
Mult_Fractions(Numerator1,Denominator1,Numerator2,Denominator2:lo
ngint;
var NumeratorOut,DenominatorOut:longint); {Umn_Drob(a,b,c,d,k,l)}
begin
  FractionReduction(Numerator1,Denominator1);
  FractionReduction(Numerator2,Denominator2);
  NumeratorOut:=Numerator1*Numerator2;
  DenominatorOut:=Denominator1*Denominator2;
end;

```

✓ процедура FractionReduction виконує скорочення дробу

```

procedure FractionReduction(var Numerator,Denominator:longint);
{Socrati(Ch, Zn)}
var A:longint;
begin
  if Denominator=0 then Denominator:=1;

```

```

    if Denominator<0 then begin Numerator:=-Numerator; Denominator:=-
Denominator end;
    a:=MinCommonDivider(Numerator,Denominator); {nod2}
    Numerator:=Numerator div A;
    Denominator:=Denominator div A;
end;

```

✓ функція MinCommonDivider знаходить спільний дільник

```

function MinCommonDivider(a,b:longint):longint; {Nod2}
begin
    if a<0 then a:=-a;
    if b<0 then b:=-b;
    while a*b<>0 do if a>b then a:=a mod b else b:=b mod a;
        result:=a+b;
end;

```

✓ процедура додавання рядків матриці Add_Rows

```

procedure
Add_Rows(adr:Adres;one,two:integer;mch1,mzn1,znak:longint);
{Sl_Strok}
var i:integer;ch1,zn1:longint;
begin
    for i:=1 to adr^.r_stl do
        begin
            Mult_Fractions(adr^.matr_ch[one][i],adr^.matr_zn[one][i],mch1,mzn1,ch
1,zn1); {Umn_Drob}

            Add_Fractions(ch1,zn1,znak*adr^.matr_ch[two][i],adr^.matr_zn[two][i],a
dr^.matr_ch[two][i],adr^.matr_zn[two][i]); {Sl_Drob}
        end;
    end;
end;

```

✓ процедура Add_Fractions додавання двох дробів:

```
procedure
Add_Fractions(Numerator1,Denominator1,Numerator2,Denominator2:longint;var
gint;var
NumeratorOut,DenominatorOut:longint);{Add_Drob(a,b,c,d:longint;var
k,l:longint);}
var r:longint;
begin
  r:=MinCommonDivider(Denominator1,Denominator2); {nod2(b,d);}
  DenominatorOut:=(Denominator1 div r)*Denominator2; {l:=(b div
r)*d;}
  NumeratorOut:=Numerator1*(DenominatorOut div Denominator1)+
  Numerator2*(DenominatorOut div Denominator2); {k:=a*(l div
b)+c*(l div d);}
  FractionReduction(NumeratorOut,DenominatorOut); {Socrati(k,l);}
end;
```

✓ Метод знаходження рангу матриці реалізується функцією
GetRang, текст якої наведений нижче.

```
function GetRang(adr:Adres;flExpert:boolean;CurMatr:TMatrix):integer;
{ 100 -продовжувати далі}
{ 1..6 -нульовий рядок }
{ 1000 -Gauss відробив }
{ 999 - Процедура була перервана за запитом користувача}
var
i,t,e,g,r:integer; ch, ch1, zn, zn1:longint;
Eto_Sistema, flExit:boolean;
s:string;
j1,j2:integer;
EdAdr:Adres;
RealNStl:integer;
iCol:integer;
LastStr:integer;
```

```

begin
  Eto_Sistema:=CurMatr.SystemEquation;
  flExit:=false;

  with adr^ do
  begin

    RealNStl:=R_stl-1;
    if Eto_Sistema then
      r:=R_str
    else r:=R_stl;
    R_stl:=r-1;

    Result:=100;

    i:=1;
    LastStr:=R_str;
    while i<=LastStr do
    begin
      if Nul_Stroka(adr,i) then
      begin
        CurMatr.DelRow(i);
        LastStr:=LastStr-1;
        if not FW_1.IsExpertDialogContinue(
          ExpertDialog4S1+' N '+IntToStr(i)+'.'+ExpertDialog4S2+'.')
        then
        begin
          flExit:=true;
          break;
        end;
        Continue;
      end;
    end;
  end;
end;

```



```

If matr_ch[i][i]=0 then
Begin
  for t:=i+1 to R_str do
    if matr_ch[t][i]<>0 then
      begin
        if FW_1.flExpert then
          begin
            Row_Exchange(adr,i,t);
            CurMatr.RefreshData;
            if not FW_1.IsExpertDialogContinue(
              ExpertDialog3S1+IntToStr(i)+'+',+IntToStr(e)) then
              begin
                flExit:=true;
                break;
              end
            end
          else
            Row_Exchange(adr,i,t);

            dop_ch:=-dop_ch;
          end;

        for e:=i+1 to r do
          if matr_ch[i][e]<>0 then
            begin
              if flExpert then
                begin
                  Col_Exchange(adr,i,e);
                  CurMatr.RefreshData;
                  if not FW_1.IsExpertDialogContinue(
                    ExpertDialog3S2+IntToStr(i)+'+',+IntToStr(e)) then
                    begin
                      flExit:=true;

```

```

        break;
    end
end
else
    Col_Exchange(adr,i,e);

    dop_ch:=-dop_ch;
end;

Result:=i;
End
Else
Begin

    for t:=i+1 to LastStr do
        if (matr_ch[t][i]<0) and (t>i) then
            begin

Mult_Fractions(matr_zn[i][i],matr_ch[i][i],matr_ch[t][i],matr_zn[t][i],ch,z
n);

                if flExpert then
                    begin
                        if (zn=1) or (ch=0) then s:=IntToStr(ch)
                        else s:=Concat(IntToStr(ch)+'/'+IntToStr(zn));

                        Add_Rows(adr,i,t,-ch,zn,1{zn}); {Str_Str_Expert(adr,i,t,-
ch,zn);}

                        CurMatr.RefreshData;
                        if not FW_1.IsExpertDialogContinue(
                            ExpertDialog3S3+IntToStr(t)+'='+ExpertDialog3S4+'
'+IntToStr(i)+'
                            '*-'+s+' '+ExpertDialog3S4+' '+IntToStr(t) ) then
                            begin

```

```

        flExit:=true;
        break;
    end

    end
else
    Add_Rows(adr,i,t,-ch,zn,zn);
end;
if flExit then break;
End;
i:=i+1;
end;

if not flExit then
    ShowMessage('Ðàñã='+inttostr(LastStr));

if not flExit then
    Result:=1000
else
    Result:=999;

R_stl:=RealNStl;

end;
end;

```

В обох основних функціях також використовуються методи класу TMatr, описані в наступному розділі:

- DelRow – видалити рядок матриці;
- RefreshData – оновити дані.

Специфіка в створенні подібних програм (опис класу)

Для візуального відображення матриці в додатку розроблений клас TMatrix. Взагалі принцип об'єктно-орієнтованого програмування, на якому базуються всі сучасні засоби розробки додатків, заснований на використанні різних класів і їхніх екземплярів — об'єктів.

Для побудови і використання об'єктів найважливіше значення мають **принципи інкапсуляції і приховування даних**. Принцип приховування даних полягає в тому, що користувачу прямий доступ до даних, як правило, заборонений. Робиться це з наступних міркувань. По-перше, для надійного функціонування об'єкта треба підтримувати **цілісність і несуперечливість** його даних (інакше користувач може занести в об'єкт такі дані, що він почне функціонувати з помилками). По-друге, необхідно ізолювати зовнішні об'єкти від особливостей внутрішньої реалізації даних. Для зовнішніх споживачів даних повинен бути доступний тільки **користувацький інтерфейс** — опис даних, а внутрішня реалізація — це справа програміста. При такому підході програміст може в будь-який момент модернізувати об'єкт, змінити структуру збереження, форму представлення даних, але, якщо при цьому не торкнутись інтерфейсу, зовнішній споживач цього не помітить і в поведженні користувача нічого не буде потрібно змінювати.

Щоб дотриматись принципу приховування даних, в об'єкті звичайно визначаються процедури і функції, що забезпечують усі необхідні операції з даними: їхнє читання, перетворення, запис. Ці функції і процедури називаються **методами**; через них і відбувається спілкування з даними об'єкта.

Сукупність даних і методів їхнього читання і запису називається **властивістю**. Крім методів, що працюють з окремими даними, в об'єкті є методи, що працюють із усією їх сукупністю.

Середовищем взаємодії об'єктів є **повідомлення**, які генеруються в результаті різних **подій**. Події відбуваються,

насамперед, внаслідок дій користувача – переміщення курсору миші, натискання кнопок миші чи клавіатури. Події можуть відбуватись також у результаті роботи самих об'єктів.

Таким чином, об'єкт можна визначити як сукупність властивостей, методів і подій, на які він може реагувати.

У Delphi частина класів представлена в графічному виді на палітрі компонентів. Клас TMatrix відрізняє від компонента відсутність реєстрації його в палітрі і, можливо, написання редактора властивостей даного потенційного компонента.

Більшість нових класів створюється спадкуванням деякого існуючого класу і розширенням його функціональних можливостей. Автори успадковували даний клас від класу TGroupBox (вікно групи).

В описі класу визначені наступні розділи:

- private (закриті) – дані, процедури і функції, визначені в цьому розділі, доступні тільки в межах даного модуля;
- protected (захищені) - процедури і функції, визначені в цьому розділі, доступні в класах нащадків;
- public (відкриті) – дані, процедури і функції доступні під час виконання (Run Time)
- published (опубліковані) - властивості і методи доступні скрізь і мають зв'язок із середовищем розробки Delphi, забезпечують виведення на екран в Інспекторі об'єктів сторінок інформації про властивості і події під час розробки (Design Time).

У розділі **private** визначені наступні дані і методи:

- FGridMatrix:TStringGrid - таблиця рядків для відображення рядків і стовпців матриці;
- FMatrDataAdr:Adres – адреса структури з даними й описом матриці;
- FMatrRowHeight:integer - висота рядків візуальної матриці;
- FMatrColWidth:integer - ширина стовпчиків візуальної матриці;

- FActive:boolean – прапор активної матриці;
- FChecked:boolean – прапор відміченої матриці;
- FSystemEquation:boolean – прапор розв'язку системи рівнянь
- FOnMatrixActivate:EMatrixActivate – подія активізації матриці;
- procedure ShowMatrix – метод виводу матриці на екран;
- procedure ShowMatrixRow(iRow:integer) – метод виводу рядка матриці з номером iRow на екран;
- procedure Show_Matrix_Item(iRow, iCol:integer) - метод виводу блоку матриці з номером рядка iRow, стовпчика iCol на екран;
- procedure SetColorActiveChecked – метод встановлення кольору активної, чи відміченої матриці;
- function GetMatrixRow:integer – метод читання властивості Row (номер активного рядка матриці);
- function GetMatrixCol:integer – метод читання властивості Col (номер активного стовпчика матриці);
- function FractionToStr(Ch,Zn:longint):String – метод перетворення чисельника і знаменника дробу в рядок);
- function FractionReduction (var Numerator, Denominator: longint): boolean – метод скорочення дробу);
- function MaxCommonDivider(a, b: longint): longint – метод визначення спільного дільника;
- procedure MatrGridDrawCell(Sender: TObject; ACol, ARow: Integer; Rect: TRect; State: TGridDrawState) – метод промальовування блоку матриці;
- procedure MatrixGridRowMoved(Sender: TObject; FromIndex, ToIndex: Integer) – метод, що виконується після переміщення рядка;
- procedure MatrixGridColumnMoved(Sender: TObject; FromIndex, ToIndex: Integer) - метод, що виконується після переміщення стовпчика;
- procedure SetMatrDataFromGrid – встановлення даних матриці з візуальної таблиці.

У розділі **public** визначені наступні методи:

- procedure SetMatrDataAdr(Value:Adres) – метод запису властивості MatrDataAdr;
- procedure SetMatrRowHeight (Value:Integer) – метод запису властивості MatrRowHeight;
- procedure SetMatrColWidth(Value:Integer) – метод запису властивості MatrColWidth;
- procedure setMatrActive(Value:Boolean) - метод запису властивості Active;
- procedure setMatrChecked(Value:Boolean) - метод запису властивості Checked;
- function GetColCount:integer - метод читання властивості ColCount;
- function GetRowCount:integer- метод читання властивості RowCount;
- procedure SetColCount(Value:integer) - метод запису властивості ColCount;
- procedure SetRowCount(Value:integer) - метод запису властивості RowCount;
- procedure SetGridColWidth – встановлення ширини в пікселях компонента по ширині таблиці рядків;
- procedure SetGridRowHeight– встановлення висоти в пікселях компонента по висоті таблиці рядків;
- function GetRealGridWidth:integer – визначення ширини таблиці рядків;
- function GetRealGridHeight:integer – визначення висоти таблиці рядків;
- procedure GridMatrixOnClick(Sender: TObject) метод, що виконується після клацання мишею по таблиці рядків;
- procedure SetSystemEquation(Value:boolean) - метод запису властивості SystemEquation.

У розділі **protected** визначені наступні методи і властивості:

- procedure RefreshData – метод відновлення даних;
- procedure ExchangeRows(Row1,Row2:integer) – метод перестановки рядків;
- procedure ExchangeCols(Col1,Col2:integer) – метод перестановки колонок;
- procedure DelRow(iRow:integer) – метод видалення рядка;
- procedure DelCol(iCol:integer) – метод видалення стовпчика;
- procedure InsRow(iRow:integer) – метод вставки рядка;
- procedure InsCol(iCol:integer) – метод вставки стовпчика;
- procedure SetFactorCol(Numerator,Denominator:integer) – метод визначення множника стовпчика;
- procedure SetFactorRow(Numerator,Denominator:integer) – метод визначення множника рядка;
- function GetFactorRow(iRow:integer):string – метод одержання множника рядка;
- function GetFactorCol(iCol:integer):string – метод одержання множника стовпчика;
- property Active:boolean read FActive write setMatrActive default true – властивість «активний»;
- property Checked:boolean read FChecked write setMatrChecked default false - властивість «відзначений»;
- property Col:integer read GetMatrixCol – властивість «номер активного стовпчика»;
- property ColCount:integer read GetColCount write SetColCount – властивість «кількість колонок»;
- property SystemEquation:Boolean read FSystemEquation write SetSystemEquation default false – властивість «система рівнянь»;
- property MatrDataAdr:Adres read FMatrDataAdr write SetMatrDataAdr – властивість «адреса даних»;

- property `MatrRowHeight:integer` read `FMatrRowHeight` write `SetMatrRowHeight` default `DefaultMatrRowHeight` – властивість «висота рядка»;
- property `MatrColWidth:integer` read `FMatrColWidth` write `setMatrColWidth` default `DefaultMatrColWidth` – властивість «ширина стовпця»;
- property `Row:integer` read `GetMatrixRow` – властивість «номер активного рядка»;
- property `RowCount:integer` read `GetRowCount` write `SetRowCount` – властивість «кількість рядків»;
- property `OnMatrixActivate:EMatrixActivate` read `FOnMatrixActivate` write `FOnMatrixActivate` – подія «активізація матриці»;
- constructor `Create(AOwner: TComponent); override` – конструктор (створює екземпляр класу);
- destructor `Destroy; override` – деструктор (знищує екземпляр класу)

Обмеження використання даної програми

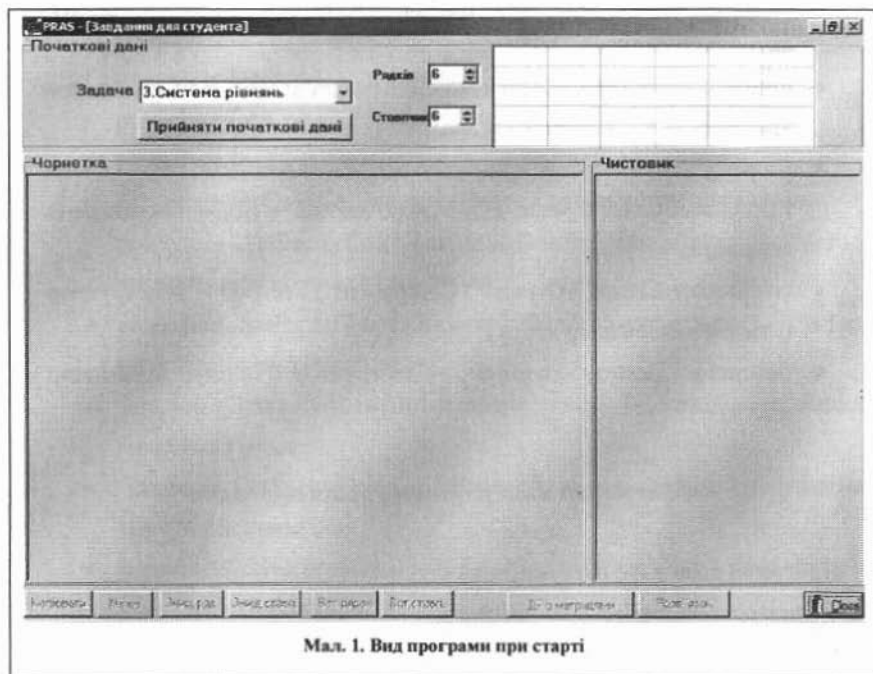
Програма розв'язує наступні задачі:

- знаходження оберненої матриці;
- визначення рангу матриці;
- розв'язання системи рівнянь методом Гаусса.

Програма працює з матрицями, у яких кількість рядків і стовпців не перевищує 6.

ПРО ЗАСТОСУВАННЯ ПРОГРАМИ (ІНСТРУКЦІЯ КОРИСТУВАЧА)

Для ініціювання виконання програми треба запустити файл **PRAS.EXE** або клацнути по іконці, у якій описаний шлях до даного файлу. Після запуску програми на екрані з'явиться вікно, зображене на Мал. 1.



Мал. 1. Вид програми при старті

Структурно форма складається з чотирьох частин. На нижній панелі розташовані функціональні кнопки. У момент запуску програми всі кнопки, крім кнопки "Close" (закрити) неактивні. Кнопки активізуються, коли вихідні дані введені і прийняті до дії (відбиваються на панелі «Чорнетка») і деактивізуються в протилежному випадку.

На верхній панелі «Початкові дані» розташована група компонентів для задання розміру матриці, введення даних у матрицю і для прийняття сформованої матриці на робочий стіл. На робочому столі (панель

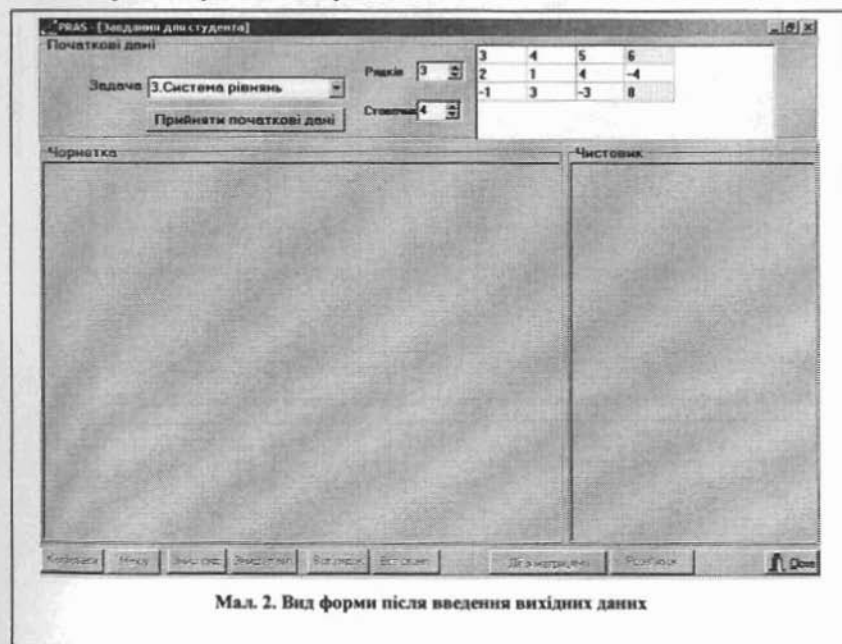
«Чорнетка») відображаються проміжні стани матриці при розв'язанні задачі. На панелі результату (панель «Чистовик») відображається підсумкова матриця.

У правому верхньому куті форми розташовані стандартні кнопки вікна:

- кнопка «Згорнути» згортає вікно до піктограми;
- кнопка «Розгорнути» розгортає вікно на весь екран;
- кнопка «Закрити» закриває вікно.

Робота в програмі починається з вибору теми задачі – візуальний компонент «Задача». Після цього встановлюється розмірність матриці (вказується кількість стовпців, колонок за допомогою візуальних компонентів «Рядків», «Стовпчиків» відповідно). Потім у таблицю рядків (візуальний компонент, що знаходиться праворуч на панелі «Початкові дані») вносяться дані матриці у виді цілих чисел чи звичайних дробів (наприклад, $1/3$).

Форма набуває вигляду, що показаний на Мал. 2.



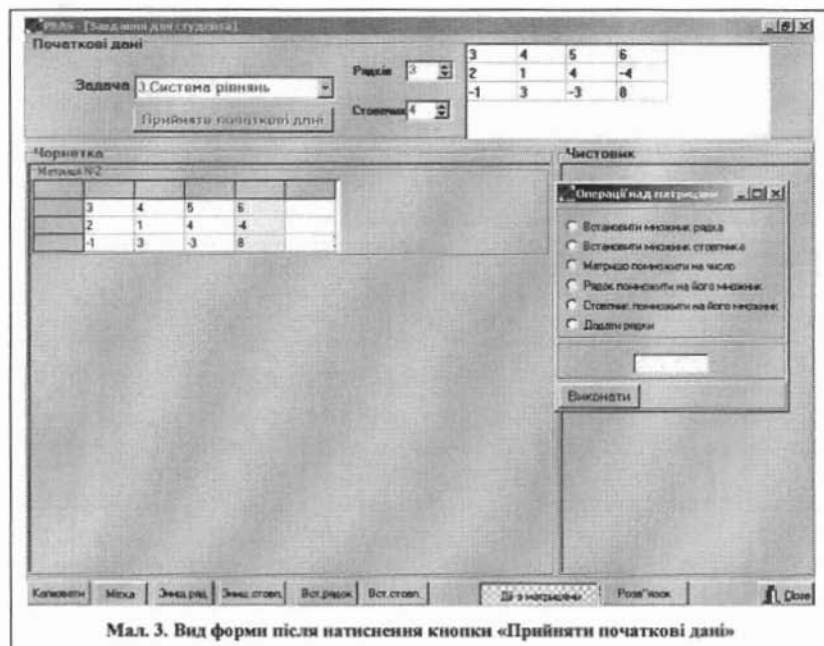
Мал. 2. Вид форми після введення вихідних даних

Після введення і перевірки правильності вихідних даних вихідна матриця поміщається на робочий стіл (панель «Чорнетка»). Операція реалізується натисканням кнопки «Прийняти початкові дані».

Після натискання цієї кнопки відбувається наступне:

- на робочому столі з'являється зображення вихідної матриці;
- активізуються всі кнопки на інструментальній панелі;
- візуалізується форма для вказання дій над матрицями.

Вид форми після натискання кнопки «Прийняти початкові дані» представлений на Мал. 3.



Для виклику експерта, що по кроках показує метод розв'язання даної задачі, потрібно натиснути кнопку «Розв'язок». При цьому на екрані з'явиться форма з текстом експерта (див. Мал. 4).

УРАС - [Виділено для студентів]

Початкові дані

Задача: Система рівнянь

Рядок: 3

Стовпець: 4

Прийняти початкові дані

3	4	5	6
2	1	4	-4
-1	3	-3	8

Матриця

Матриця A:

3	4	5	6
2	1	4	-4
-1	3	-3	8

Матриця B:

Чистовик

Оформити результат

☐ Встановити нульові рядки

☐ Встановити нульові стовпці

☐ Матрицю перетворити на число

☐ Рядки перетворити на його нульові

☐ Стовпці перетворити на його нульові

☐ Видалити рядки

Виконати

Експерт

Для побудови оберненої матриці необхідно створити допоміжну матрицю і вкодувати в неї одиницю. Для цього необхідно створити матрицю розміром 2x4. Кожний елемент матриці вводити по одному. Кожний елемент перетворюється на обернену до одиниці. Я буду виконувати цю роботу по кроках.

Користувач

OK

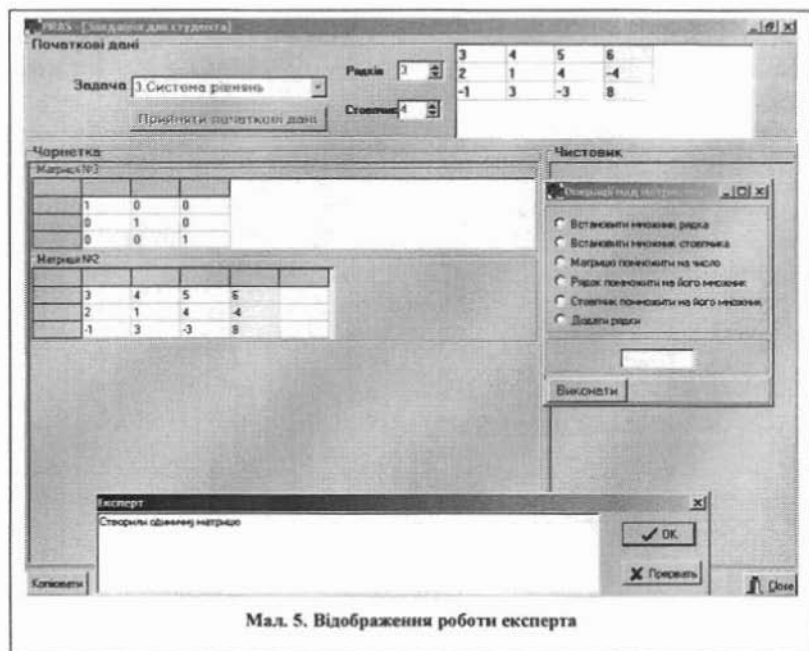
Продовжити

Довідка

Мал. 4. Форма з текстом експерта

Ініціація виконання наступного кроку розв'язання задачі досягається натисканням кнопки «ОК», переривання дії експерта здійснюється натисканням кнопки «Перервати».

У процесі кожного кроку розв'язання задачі експертом на робочому столі графічно зображується останній стан даних, що відповідає стану даних після виконання дій даного кроку (див. Мал. 5) .

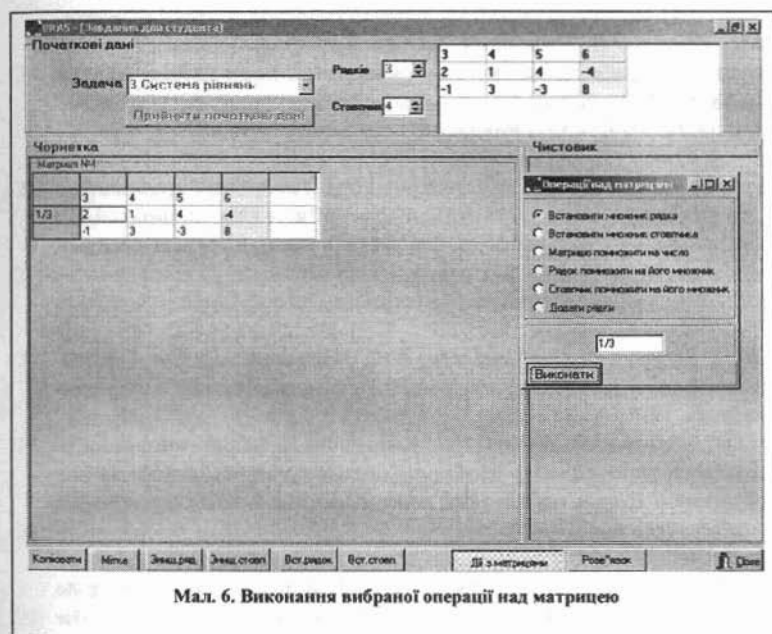


Після останнього кроку розв'язання задачі експерт зникає з екрана.

Для ручних операцій над матрицею її треба активізувати, клацнувши мишкою по будь-якій області необхідної матриці. Заголовок матриці при цьому зафарбується в червоний колір.

Потім на формі операцій над матрицями потрібно вибрати з переліку операцій необхідну (клацнувши мишкою по потрібному тексту), якщо видимий компонент введення даних, ввести необхідні дані і натиснути кнопку «Виконати», що ініціює виконання обраної операції (див. Мал. 6).

Форму операцій над матрицями можна сховати, натиснувши кнопку «Дії з матрицями». Візуалізується ця форма натисканням цієї ж кнопки.



Література

1. *Алексеев Е.А., Довгяло А.М., Платонов Б.А., Платонова О.П.* Система программирования обучающих курсов. – К.: Выща шк., 1981. – 239 с.
2. *Битцер Д., Джонсон Р.* Система PLATO – техническое средство обучения, использующее ЭВМ // Кибернетика и педагогика. – М., 1972. – С.147-156.
3. *Битцер Д., Ислей С.* Система обучения, управляемая ЭВМ (PLATO) // Зарубежная электроника. – 1967. — №10. – С. 66-78.
4. *Битцер Д., Скапердаз Д.* PLATO-IV – экономичная обучающая система, управляемая мощной ЦВМ // Зарубежная электроника. – 1969. – № II. – С.46-54.
5. *Бондаренко В.К.* Технические средства обучения. – М.: Знание, 1978. – Вып. 6. – 47 с.
6. *Брановицкий В.И., Гецко Л.Н.* Вопросы организации генерирующих обучающих систем / на базе ППП ПРОЛОГ-ЕС // Управл. машины и системы. – 1983. – № 4. – С. 75-80.
7. *Будаунаскене К.Т.* Генерирование заданий при помощи ЭВМ для системы контроля знаний // ЭВМ в учебном процессе и управлении вузами. – Вып. I. – Рига, 1973. – С. 38-42.
8. *Довгяло А.М., Небрат О.П., Платонов В.А.* Обучение с использованием вычислительных машин: Современное состояние и перспективы // Управляющие системы и машины. – 1978. – № 2. – С.2-20.
9. *Довгяло А.М., Ющенко Е.Л.* Обучающие системы нового поколения // УСМ. – 1988. – № 1. – С. 83-86.
10. *Гладышевский Н.Н., Михайлов А.И., Огородников Ю.А.* Технические средства обучения. — М.: Отд. науч. информ. НИИВШ, 1979. – 40 с. – Обз. информ. / НИИ пробл. высш. школы. Отд. науч. информ. Сер. Обуч. в высш. и сред. школе; вып. 3.
11. *Гребень И.И., Довгяло А.М.* Автоматические устройства для обучения. – К.: Изд-во КГУ, 1965. – 197 с.
12. *Жегалин В.А., Первин Ю.А., Хоперсков А.Е.* Генератор задач – учебно-ориентированный пакет прикладных программ // Прикладные методы информатики. - Новосибирск, 1980. – С. 112-120.
13. *Журавлев О.В., Сороко В.Н., Чертов О.Р.* Концепции и метод автоматизации блочного проектирования структур проблемно-ориентированных систем для машинного обучения // Вторая научно-техн. конф. советских и польских молодых ученых – выпускников высш. учеб. заведений СССР, Киев, 1986. – Вроцл. политех. ин-т, 1986. – С. 219-222.
14. *Журавлев О.В., Сороко В.Н., Чертов, Захаревич К.Г.* Методические указания по составлению контролирующих курсов и технологии работы в системе ДИСК/МИКРО – К.: УМК Минвуза УССР, 1989 – 36с.
15. *Зайцева Л.В., Ниццкий Л.В., Новицкий Л.П., Шитиков В.С.* Диалоговая система обучения и контроля знаний КОНТАКТ / ОС на базе ЕС ЭВМ // Диалоговые системы. – 1981. - № 4. – С. 87-103.
16. *Замов Н.К., Зискин В.Ф., Пишеничный П.В.* Обеспечение работы автора в автоматизированной обучающей системе Гамма // Исследования по прикладной математике. – Изд-во Казанск. ун-та, 1980. – Вып. 8. – С. 99-104.

17. Ковалев В.А. Метаязыковые средства обучения в диалоговых системах. // Кибернетика и вуз: Межвуз. научно-техн. сб. – Томск, 1980. – Вып. 15. – С. 152-154.
18. Корнейчук В.И. Методические указания и система тестов для программированного контроля знаний по ЦВМ. – К., 1977. – 47 с.
19. Корнейчук В.И., Пиксотов В.В., Сороко В.Н. Метод диагностики знаний при программированному контролю // Підвищення ефективності використання технічних засобів навчання. – К.: Київськ. держ. пед. ін-т, 1981. – С.77-80.
20. Корнейчук В.И., Пиксотов В.В., Сороко В.Н., Марковский А.П. Вопросы построения информационно-методического обеспечения обучающих систем // Вестник КПИ, Научно-метод. сер., 1982. – Вып.6. – С 57-64.
21. Корнейчук В.И., Сороко В.Н., Журавлев О.В., Захаревич К.Г. Вопросы развития структур и функциональных компонент АОС // Методы и средства кибернетики в управлении учебным процессом высшей школы. – Рига: Рижск. политехн. ин-т, 1985. – С.138-151.
22. Корнейчук В.И., Сороко В.Н., Журавлев О.В., Чертов О.Р. Метод автоматизации проектирования структур АОС, использующий средства логического вывода. // Эффективность применения автоматизированных обучающих систем в учебном процессе высшей школы: Тез. докл. Всесоюзн. научно-метод. совещания, Рига, 1988. – Рига: РПИ, 1988. – С.13-15.
23. Корнейчук В.И., Сороко В.Н., Журавлев О.В., Чертов О.Р. Метод и средства автоматизации проектирования технического и программного обеспечения АОС // Методы и средства кибернетики в управлении учебным процессом высшей школы. – Рига: РПИ, 1989. – С.18-26.
24. Корнейчук В.И., Сороко В.Н., Захаревич К.Г. Машинная генерация учебных контрольных заданий в АОС // Управляющие системы и машины. – 1985. – № 2. – С. 8-122.
25. Корнейчук В.И., Сороко В.Н., Захаревич К.Г. Способ машинного представления информационно-методического обеспечения АОС // Вестник КПИ, Научно-метод. сер., 1989. – Вып.13. – С. 38-42.
26. Корнейчук В.И., Сороко В.Н., Пиксотов В.В., Тодоров Г.С. и др. Метод составления равноценных вариантов учебных заданий // Вопросы педагогики и частных методик. – Вып. 2. – Уфа: УАИ, 1978. – С.79-86.
27. Корнейчук В.И., Тодоров Г.С., Сороко В.Н. О программированном контроле знаний с помощью ЦВМ // Программированное обучение. – К., 1978. – Вып. 15. – С. 92-97.
28. Круг Г.К., Кабанов В.А., Черных А.В. Инструментальные диалоговые обучающие системы на базе микроЭВМ // Микропроцессорные средства и системы. – 1987. – №3. – С. 29-30.
29. Лаптев О.Н. Основные принципы машинной генерации вариантов контрольных заданий // Проблемы комплексного использования вычислительной техники в народном образовании. – Свердловск: Свердл. гос. пед. ин-т. – 330. – 1980. – С.79-92.
30. Лаптев О.Н. Система автоматизированной проверки знаний. Методическая разработка. – Свердловск: Свердловск. пед. ин-т, 1985. – 67 с.

31. Лобанов Ю.М., Васильев А.В., Токарева В.С. и др. Учебно-методическое обеспечение автоматизированных обучающих систем. – М.: НИИВШ. – Вып.6. – 1987. – 40 с.
32. Львов М.С., Спиваковский А.В. Введение в объектно-ориентированное программирование: Учебное пособие. — Херсон:Айлант, 2000. — 210с.
33. Львов М.С., Спиваковский О.В. Основи алгоритмізації та програмування: Навч. посібник. — Херсон: Айлант, 2000. — 214с.
34. Методические указания по формированию и использованию машинных контролирующих курсов для обучающих систем на базе ЕС ЭВМ / Сост. Слипченко В.Г., Сороко В.Н., Захаревич К.Г. и др. – К.: ГИВЦ Минвуза УССР, 1985. – 38 с.
35. Маляров А.Н. Техническое обеспечение автоматизированных обучающих систем на базе микропроцессоров // Проблемы программированного обучения. Материалы УП советско-французского семинара. – М.: Изд-во МЭИ, 1981. – С. 46-56.
36. Мизинцев В.П. Метод расчета учебной информации на основе графовых моделей // Программированное обучение. – К., 1976. – Вып. 13. – С.8-17.
37. Михалевич В.М., Довгялло А.М., Савельев А.Я., Когтов Н.М. Экспертно-обучающие системы и комплексы компьютерных средств обучения // Современная высшая школа. – 1988. - № 1 /61/. – С. 125-135.
38. Минский М. Фреймы для представления знаний. – М.: Энергия, 1979. – 152 с.
39. Никитин А.В., Гладышевский Н.Н. Современное состояние и перспективы применения технических средств обучения и контроля. – М.: НИИВШ, 1976. – 81 с.
40. Ниццкий Л.В., Новиков В.А. Технические и методические требования к автоматизированным обучающим системам на базе ЕС ЭВМ. – Рига: РПИ, 1979. – 34 с.
41. Новиков В.А. Направления развития автоматизированных обучающих систем // Сборник науч.-метод. статей по ТСО. – М.: Высш. шк., 1978. – Вып. 2. – С.7-10
42. Новиков С.В., Шумляев В.С. Программное обеспечение автоматизированной обучающей системы. – Изд-во БГУ, 1982. – 143 с.
43. Новицкий Л.П., Фельдберг Л.М. Экспертно-обучающая система для персональной ЭВМ // Методы и средства кибернетики в управлении учебным процессом высшей школы. – Рига: РПИ, 1989. – С. 27-31.
44. Обучающие машины, стистемы и комплексы: Справочник / Под общ. ред. Савельева А.Я. – К.: Выща шк., 1986. – С. 301-304.
45. Попов Э.В. Классификация экспертных систем и инструментальных средств // Моделирование и искусственный интеллект: Сб. науч. тр. – М.: МИРЭА, 1988. – С. 23-33.
46. Паскин Е.Н., Митин А.И. Автоматизированная система обучения ЭКСТЕРН. – Изд-во МГУ, 1985. – 145 с.
47. Построение экспертных систем /Под ред. Ф.Хейес-Рота, Д.Уотермана, Д.Лената. – М.: Мир, 1987.
48. Проблемы применения ЭВМ в высшем образовании за рубежом. – М., 1973. – 64 с. - /Реф. информ./ИЦВШ, Сер. Актуальные проблемы учебного процесса/.
49. Рамиль Альварес Х. Организация диалога в микропроцессорной АОС // Диалоговые компьютерные системы. – Изд-во МГУ, 1986. – С. 80-84.
50. Савельев А.Я. Автоматизированные обучающие системы на базе ЭВМ. – М.: Знание, 1977. – Вып. I. – 34 с.

51. Савченко А.В., Филиппов А.Н., Федоров П.И. и др. Разработка автоматизированных обучающих систем на базе микроЭВМ в Московском институте электронной техники. – М., 1987. – 56 с.
52. Слипченко В.Г., Сороко В.Н., Захаревич К.Г. и др. Методические указания к созданию обучающее-контролирующих курсов к использованию генерирующей АОС «ДИСК» - К.: ГИВЦ Минвуза УССР, 1981. – 60 с.
53. Сороко В.Н., Кунцевич М.В. О Структурном анализе автоматизированных обучающих систем // Управл. сист. и машины. – 1982. - № 1. – С.97-102.
54. Співаковський О.В., Крекнін В.А. Лінійна алгебра. Навч.посібник. — Херсон, 1998.— 144 с.
55. Талызина Н.Ф. Теоретические проблемы программированного обучения. – М.: Изд-во МГУ, 1969. – 132 с.
56. Уотерман Д. Руководство по экспертным системам. – М.: Мир, 1989.
57. Шрейдер Ю.А. Экспертные системы; их возможности в обучении // Вестник высшей школы. – 1987. - №2. – С.14-19.
58. Эффективность применения автоматизированных обучающих систем в учебном процессе / Тез. докл. на Всесоюз. научно-метод. совещ., 2-4 марта 1988 г. – Рига: Изд-во РПИ, 1988. – 202 с.

ЗМІСТ

1. Вступ (основні задачі програми)	2
2. Погляд на створення подібних програм	7
3. Опис структури програм	10
4. Про застосування програми (інструкція користувача).....	33
5. Література	39