**Міністерство транспорту та зв’язку України**

**Державний департамент з питань зв’язку та інформатизації України**

**Одеська національна академія зв’язку ім.О.С.Попова**

***Курсова робота***

***з предмету***

**“Програмування та алгоритмічні мови”**

тема:

***„Програмування в Delphi”***

Виконав(ла) студент(ка) гр. САТкв-37:

Носур І.

перевірив: Власенко В.М.

з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2007**

**ЗМІСТ**

Вступ

ТЕОРЕТИЧНА ЧАСТИНА

Програмування в Delphі.

Основні елементи програми.

Основні частини програми.

Особливості архітектури програми в ос windows.

Архітектура програми в ос windows, взаємодія програми з операційною системою.

Тест-програма.

ПРАКТИЧНА ЧАСТИНА

Метод гаусса

Висновок

Список використаної літератури

Додаток

**ВСТУП**

Метою даної курсової роботи є розробка програми використання метода Гаусса для ПЕОМ типу PC з операційною системою Windows. Розробка програми повинна вестись за допомогою мови програмування Object Pascal в середовищі Delphi – для операційної системи Windows 9x-XP.

В даній курсовій роботі викладено теоретичний матеріал для даного метода, алгоритм роботи програми, текст та зовнішній вигляд програм.

Розроблену програму можна використовувати в навчальному процесі.

**ТЕОРЕТИЧНА ЧАСТИНА**

**Програмування в Delphi.**

Delphi - це об’ектно-орієнтоване середовище для візуальної побудови програмних продуктів, засноване на мові Object Pascal, що є переробленою й істотно доповненою версією Turbo Pascal фірми Borland.

Програмування в Delphi складається з двох основних етапів:

1. Візуальна побудова програми на основі об’єктних компонентів і настроювання їх властивостей, в результаті чого можна швидко сформувати користувальницький інтерфейс і забезпечити значну частину функціональності додатка.
2. Написання програмного коду мовою Object Pascal для забезпечення особливої функціональності додатка, яку неможливо досягти використанням візуальної побудови.

З моменту першої реалізації мови Pascal технології програмування зробили величезний крок вперед. І, незважаючи на те, що Pascal теж розвивався, його останній версії, випущеній фірмою Borland у 1990 році, істотно бракує можливостей, властивих іншим сучасним мовам програмування. В середовищі Delphi використовується оновлений варіант цієї популярної мови. Зміни торкнулись, перш за все, об’єктно- орієнтованої частини Pascal, однак, здавалося б цілком завершені інструменти структурного і модульного програмування також придбали деякі нові особливості.

Розглянемо основні елементи мови програмування Pascal, звертаючи увагу на зміни, що відбулися з ним в Delphi.

**Основні елементи програми**

Основними елементами програми мовою Object Pascal є:

* оператори - команди, що визначають структуру програми (наприклад, оператори розгалуження і зациклення) або виконуючі які-небудь дії (наприклад, арифметичні операції чи операції порівняння):
* ключові слова - команди, використовувані звичайно для відокремлення одних частин програми від інших. Ключові слова призначаються для компілятора, губляться в процесі побудови програми, що виконується і не виконуються в буквальному значенні цього слова;
* директиви компілятору - аналогічні ключовим словам, але впливають не на програму і її структуру, а на процес компіляції і побудови програми. Частину директив компілятору можна установити за допомогою головного меню інтегрованого середовища, однак використання директив у тексті програми допомагає використовувати їх вибірково - не для всієї програми, а для окремих її частин.

**Основна частина програми**

Програма на Object Pascal, як і в більш ранніх версіях Pascal, складається з основної частини, власне і названою програмою (англ. Program - Програма) і декількох модулів (англ. Unit - Модуль), що підключаються на етапі компіляції до основної частини програми, наявність яких не обов'язкова. Розглянемо структуру основної частини програми.

Program

.......... Заголовок програми

Uses

........... Розділ підключення модулів (бібліотек)

Label

··········· Розділ опису міток безумовного переходу Const

........... Розділ опису констант

Type

........... Розділ опису нестандартних типів даних

Var

........... Розділ опису змінних

……....Початок розділу опису логіки програми

Begin

........... Розділ опису логіки програми

End. Кінець розділу опису логіки програми

Основна частина програми міститься в окремому файлі і складається з одного чи декількох розділів, кожний з який починається з визначеного ключового слова (Program, Uses, Label і т.д.). Якщо в існуванні розділу немає необхідності, то ключове слово, що його відкриває, не вказується. Основна частина програми в Delphi має розширення dpr (від aнгл. DPr - Delphi Project - Проект Delphi) і, у більшості випадків, не вимагає втручання програміста. Створення і відновлення основної частини програми бере на себе середовище розробки.

З усіх представлених розділів обов'язковим в Pascal є тільки розділ опису логіки програми (починається ключовим словом Begin, закінчується ключовим словом End із крапкою). У програмі на Delphi обов'язковим є також заголовок програми, що має вид:

Program <Назва>;

Вимога до наявності заголовка програми зв'язана з посиланнями на файли ресурсів, що підключаються до програми, і назви, що мають такі ж, що і файл, у якому зберігається основна частина програми. Дані посилання додаються середовищем автоматично, а виправлення їх вручну не рекомендується, тому що це може викликати порушення структури проекту.

Назва програми вибирається за звичайними правилами іменування ідентифікаторів у Turbo Pascal. Ехе- файл, одержаний в результаті компіляції програми, має ім'я, що відповідає назві файлу з програмою.

Розділи опису констант (починається ключовим словом Const), нестандартних типів даних (ключове слово Туре) і змінних (ключове слово Var) можуть знаходитись один за одним у будь-якому порядку, однак приведений порядок є найбільш прийнятним. Це пов'язано з тим, що константи можуть використовуватися при описі типів даних і змінних, а типи даних звичайно використовуються при описі змінних. При цьому ідентифікатори, описані в розділі опису змінних, ні при яких умовах не можуть використовуватися в розділах опису типів і констант. Область дії ідентифікатора починається з моменту його опису, тобто використання ідентифікатора можливо тільки нижче по тексту програми.

**Модулі**

Модуль (бібліотека), так само як і основна частина програми, міститься в окремому файлі і складається з декількох розділів, аналогічних розділам основної частини програми. Розглянемо структуру модуля.

Unit Заголовок модуля

interface Вказівка на початок интерфейсной секції

Uses

.......... Розділ підключення модулів (бібліотек)

Const

.......... Розділ опису констант

Type

........ Розділ опису типів даних

Var

......... Розділ опису перемінних

Implementation Вказівка на початок описової секції Uses

........... Розділ підключення модулів

Label

........... Розділ опису міток безумовного переходу

Const

........... Розділ опису констант

Турі

........... Розділ опису типів даних

Var

........... Розділ опису змінних

initialization Початок секції ініціалізації

......... . Розділ опису логіки ініціалізації

finalization Початок секції деініціалізації

........... Розділ опису логіки деініціалізації

End. Закінчення модуля

Вихідний текст модуля може містити чотири секції:

* ***Інтерфейсна секція*** (починається ключовим словом interface), в якій розташовуються заголовки процедур і функції, а також опису констант (розділ опису констант починається з ключевого слова Const), нестандартних типів даних (ключове слово Туре) і змінних (ключове слово Var). Всі ідентифікатори, описані в интерфейсной секції, доступні для використання підключаючим (модуль) частинам програми, поряд з їхніми власними описами.
* ***Описова секція*** (починається ключовим словом Implementation), в якій розташовуються описи процедур і функції, заголовки яких зазначені в інтерфейсній секції, а також інші процедури і функції, використовувані підпрограмами даного модуля, але недоступні фрагментам програм, що підключають модуль. Також в описовій частині модуля можуть розташовуватися описи міток безумовного переходу, констант, нестандартних типів даних і змінних (починаються, відповідно, із ключових слів Label, Const, Туре і Var). Усі ці описи також не доступні фрагментам програми, що подключили модуль, і використовуються тільки для внутрішніх цілей бібліотеки.
* ***Секція ініціалізації*** (починається ключовим словом initialization), що містить команди, які необхідно виконати при підключенні модуля до програми. Таким чином, перед виконанням якої-небудь програми, до якої підключаються зовнішні модулі, спочатку виконуються ініціалізовані частини модулів (у порядку підключення). Дана секція не є обов'язкової.
* ***Секція деініціалізації,*** використовується тільки при наявності секції ініціалізації, що містять команди, які необхідно виконати при завершенні додатка. Дана секція не є обов'язкової і може використовуватися для звільнення ресурсів, зайнятих у секції ініціалізації.

Наявність ключового слова End із крапкою є обов'язковим і означає закінчення модуля.

Наявність інтерфейсної і описової секцій у бібліотеці є обов'язковим, навіть, якщо в цих секціях нічого не міститься. Таким чином, ключові слова Interface і Implementation повинні бути присутніми в тексті модуля завжди.

Назва модуля - ідентифікатор, що вказується після ключового слова Unit у заголовку модуля. Назва модуля будується за звичайними правилами іменування ідентифікаторів і повинна відповідати імені файлу, в якому знаходиться опис модуля. Саме за назвою здійснюється підключення модулів до основної частини чи програми до інших модулів. Як уже згадувалося раніше, підключення виконується в розділі підключення модулів Uses за назвою.

Розглянемо приклад найпростішого модуля і програми, що підключає до себе цей модуль. Код модуля приведений в лістінгу 5.1, а код програми, що його підключає - в лістінгу 5.2.

Лістінг 5.1. Найпростіший модуль

Unit Unitl; {Заголовок модуля, назва модуля - Unit1}

Interface {Вказівка на початок інтерфейсної секції}

Implementation {Вказівка на початок описової секції}

End. {Закінчення модуля)

Лістінг 5.2. Найпростіша програма, що підключає модуль

Program Program1; {Заголовок програми,

назва програми - Program1} Uses {Розділ підключення модулів}

Unitl; {Підключення модуля Unitl}

Begin (Початок розділу опису логіки програми}

End. (Кінець розділу опису логіки програми)

**Проект**

Важливим нововведенням в Delphi є поняття проекту - опису частин, на які розбитий вихідний текст програмного продукту. Крім файлів з вихідними текстами, у проект можуть входити файли ресурсів, що підключаються до програми (наприклад, зображення чи тексти).

Проект призначений для підвищення зручності розробки програмних продуктів, особливо великих, тому що середовище розроблювача має можливість контролювати елементи проекту, наприклад, для підтримки актуальності відкомпільованих частин програми.

Опис проекту знаходиться в основній частині програми, генерується середовищем автоматично, і в більшості випадків, не вимагає втручання програміста. Основна частина програми знаходиться у файлі з розширенням dpr і містить перерахування всіх модулів, що входять в npoekт, а також ініціалізовану частину програми і деяку допоміжну інформацію для більш тонкого настроювання середовища під конкретний проект.

**Особливості архітектури програми в ос windows**

Об’ектно-орієнтоване програмування - це не просто сучасна технологія створення програмних продуктів, а новий загальновизнаний спосіб мислення. Об’ектно-орієнтований підхід істотно знижує складність програмних продуктів, що дозволяє розроблювачам мов програмування з самого початку закласти в них можливості рішення чималої кількості широко розповсюджених задач. Причому це стосується практично всіх аспектів реалізації програм, у тому числі й алгоритмічних частин, і фрагментів програм, керуючих даними.

Оскільки створення програмних продуктів, призначених для роботи під керуванням операційної системи Windows, також має істотні особливості, розуміння яких особливо необхідно для більш чіткої організації роботи програми , то на початку розглянемо саме їх.

**Архітектура програми в ос Windows. Багатозадачність**

Операційна система Windows є багатозадачною, що припускає досить тверді вимоги до поводження програм. Одна із самих істотних вимог - неможливість монопольного використання ресурсів, тому що додаток не може самостійно визначити, хто саме в даний момент може використовувати ресурс, і в яку саме програму повинна потрапити інформація, наприклад, введена користувачем із клавіатури.

У зв'язку з цим у Windows реалізується схема програм, керованих подіями. Це значить, що кожна програма являє собою набір підпрограм, деякі з який викликаються безпосередньо ядром операційної системи для реакції на яку-небудь подію, що виникла в системі. При цьому жодний додаток не може чекати здійснення події, монополізуючи, таким чином, ресурси і припиняючи роботу інших додатків.

Аналізом змін у системі і розсиланням повідомлень про події займається безпосередньо операційна система. Структура програми в Windows представлена на рис. 5.1.



Рис. 1. Загальна структура програми в Windows

**Багатовіконність**

Кожній програмі в Windows зіставляється вікно — деяка область на екрані, в якій можна виводити інформацію. Вікно може займати весь екран, чи тільки його частину, воно може перебувати за межами екрана, бути згорнуте (сховане) за бажанням користувача чи перекрито вікнами інших додатків.

Програма може визначити розташування свого вікна на екрані, хоча звичайно в цьому немає необхідності. Вся інформація, виведена програмою, буде автоматично розташована в її вікні. Причому центр координат (точка з нульовими координатами), використовуваних при виведенні, знаходиться в лівому верхньому куті вікна, тобто не залежить від розташування вікна на екрані.

Віконне представлення програм відповідає умовам, що накладається на систему багатозадачністю, тому що кілька додатків одночасно не можуть виводити свою інформацію в те саме місце.

Крім зручного представлення інформації для користувача, вікно має другу важливу функцію з погляду програми - в будь-який момент часу одне з вікон «знаходиться в фокусі» (є «активним») і потоки інформації, що вводяться, направляються операційною системою в програму, якій належить це вікно. Механізм такого перенапрямку потоків введення заснований на подіях. Розглянемо докладніше структуру програми в Windows.

**Програма, керована подіями**

Windows стежить за змінами в стані системи (натискання клавіш, переміщення миші і натискання її кнопок, і т.д.) і генерує спеціальні структури даних, що містять наступну інформацію:

1. Посилання на вікно, що знаходилося в фокусі, коли відбулася подія.
2. Ідентифікатор події Message (значення типу Word).
3. Додаткова інформація для даної події (два значення - одне типу Word, друге типу Longlnt), наприклад, код натиснутої клавіші для повідомлення про натискання клавіші, чи координати миші для повідомлення про переміщення миші.

Далі, ці пакети зберігаються в спеціальній черзі повідомлень Windows. а ядро знову переходить в режим спостереження за системою. З іншого боку, черга безупинно розбирається, аналізується посилання на вікно, якому призначена подія, і викликається його віконна функція - спеціальна підпрограма, описана за визначеними правилами і зареєстрована в операційній системі як об’єкт призначення для повідомлень. Заголовок віконної функції має такий вигляд:

Function WindowProc(Window: HWnd; Message, WParam: Word: LParan: Longint): Longint; export;

Життєвий цикл повідомлення представлений на рис. 2.

Викликана віконна функція повинна проаналізувати подію по її ідентифікатору Message, використовуючи додаткові параметри WParam і LParam, виконати дії, обумовлені логікою програми, і повернути як результат функції код завершення обробки.



Рис. 2. Повідомлення в Windows

Якщо додаток не може проаналізувати код події (обробка такої події не передбачена у віконній функції), то повинний бути викликаний оброблювач за замовчуванням. Таким чином, велику частину віконної функції звичайно займає оператор множинного вибору Case, що на підставі інформації про повідомлення виконує необхідні дії. У секції Else такого оператора викликається оброблювач за замовчуванням (див. рис. 3).



Рис. 3. Обробка повідомлення у віконній функції

Вікно, що створюється системою, повинне належати деякому класу, зареєстрованому в ній. Під час реєстрації системі передається інформація про х параметри вікна, наприклад, про набір його візуальних елементів (наявність меню, колір і стиль фону, і т.д.), а також посилання на віконну функцію, екземпляр якої буде створений для кожного вікна, належного даному класу.

Розглянемо основні етапи роботи програми в ОС Windows:

1. Створення і реєстрація класу вікна, передача операційній системі посилання на віконну функцію.
2. Створення вікна на основі класу.
3. Організація попереднього аналізу черги повідомлень, робота віконної функції, визваної ядром Windows.
4. Знищення вікна, дерегістрація класу вікна.

У такий спосіб Windows-програма, яка нічого не робить, займає порядком сімдесятьох рядків і надає безліч можливостей для появи в ній помилок. При цьому переважна більшість додатків виконують ті самі дії при обробці повідомлень, та й набір оброблюваних повідомлень приблизно однаковий. У такій ситуації логічне рішення про інкапсуляцію характеристик вікна і його подій в деякий клас, що містить методи (звичайно порожні), визваних при обробці подій.

Дані методи доступні для перевизначення і надають можливості для зручної роботи з вікном, засновані на логічному представленні вікна в екземплярі відповідного класу, а не на викликах системних функцій.

Приховання складних процесів, що відбуваються з додатком у Windows і об’єктно-орієнтований підхід до програмування - основний напрямок розвитку мов програмування в даний час. Виключенням не став і Delphi. Всі процеси, зв'язані з підтримкою працездатності віконної програми, реалізовані розроблювачами мови. Це дозволяє програмісту зосередитися на розробці логіки програми, а не тієї її частини, що взаємодіє з системою.

**Взаємодія програми з операційною системою**

Програмування під DOS зв'язано з частим звертанням до ресурсів, наприклад, до відеокарти, прямо без виклику функцій операційної системи, що підвищує швидкодію програм за рахунок відмовлення від універсальності. Однак такий підхід до програмування в Windows неможливий, тому що може порушити цілісність інших додатків.

Ще один недолік прямого підходу складається в необхідності самостійної роботи з пристроями, що іноді істотно відрізняються один від одного, особливо в даний час. В Windows ця проблема вирішена за допомогою універсального програмного інтерфейсу операційної системи, що забезпечує однаковий доступ до пристроїв одного типу за допомогою набору так званих API-функцій (від англ. API - Application Programming Interface - інтерфейс прикладного програмування). Наприклад, незалежно від відеокарти, встановленої на конкретному комп'ютері, для виведення зображення використовуються ті самі функції.

Незважаючи на зниження швидкодії, визване більш складним шляхом при виконанні операції «програма - операційна система -драйвер - пристрій», програма стає абсолютно переносною, тобто може бути запущена на комп'ютері з будь-якою конфігурацією.

Таким чином, Windows-програми значно тісніше зв'язані з операційною системою, чим програми, що працюють у DOS.

**Програма-тест**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Forms,Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Memo1: TMemo;

Edit1: TEdit;

Button1: TButton;

Label1: TLabel;

Label2: TLabel;

Button2: TButton;

Button3: TButton;

Memo2: TMemo;

Label3: TLabel;

Label4: TLabel;

Edit2: TEdit;

Label5: TLabel;

Edit3: TEdit;

Label6: TLabel;

procedure FormCreate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

Var

k,km,m : Integer;

Sk,Skm : String;

procedure TForm1.FormCreate(Sender: TObject);

begin

Memo1.Clear;

Memo2.Clear;

Edit1.Clear;

Edit2.Clear;

Edit3.Clear;

Memo1.ReadOnly := True;

Memo2.ReadOnly := True;

Edit2.ReadOnly := True;

Edit3.ReadOnly := True;

k := 1;

m := 0;

Label3.Caption := '';

Label3.Caption := 'Основні елементи програми:';

Memo1.Lines.Add('1. Дійсні');

Memo1.Lines.Add('2. Цілочисельні');

Memo1.Lines.Add('3. Літерні');

Memo1.Lines.Add('4. Оператори, ключові слова, директиви компілятору');

Memo1.Lines.Add('5. Логічні');

end;

procedure TForm1.Button1Click(Sender: TObject);

Var

i : Integer;

Si : String;

begin

Si := Edit1.Text;

i := StrToInt(Si);

Case k of

1: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

2: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

3: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

4: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

end;

5: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

6: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

7: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

8: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

9: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

10: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

11: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

12: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

13: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

end;

14: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

4: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

15: Case I of

1: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

2: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

3: Begin

Memo2.Lines.Add (' Відповідь вірна');

m := m + 1;

End;

4: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

5: Begin

Memo2.Lines.Add (' Відповідь не вірна');

Memo2.Lines.Add ('Опрацюйте матеріал спочатку');

End;

end;

end;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

Memo1.Clear;

Memo2.Clear;

Edit1.Clear;

Label3.Caption := '';

k := k + 1;

Case k of

2: Begin

Label3.Caption := 'З чого складається програма ?';

Memo1.Lines.Add('1. StringAnsi');

Memo1.Lines.Add('2. StringShort');

Memo1.Lines.Add('3. З основної частини і декількох модулів');

Memo1.Lines.Add('4. Short');

Memo1.Lines.Add('5. WideShort');

End;

3: Begin

Label3.Caption := 'Яким ключовим словом починається заголовок програми?';

Memo1.Lines.Add('1. Статистичні і динамічні');

Memo1.Lines.Add('2. Формальні і статичні');

Memo1.Lines.Add('3. Типові і динамічні');

Memo1.Lines.Add('4. Program');

Memo1.Lines.Add('5. Логічні і формальні');

End;

4: Begin

Label3.Caption := 'Чим починається і закінчується розділ опису логіки програми?';

Memo1.Lines.Add('1. Об'+'"'+'єднання множин : +');

Memo1.Lines.Add('2. Пересічення множин : \*');

Memo1.Lines.Add('3. Еквівалентність : =');

Memo1.Lines.Add('4. Перевірка входження : <=');

Memo1.Lines.Add('5. Begin..End.');

End;

5: Begin

Label3.Caption := 'Який розділ програми є обовязковим?';

Memo1.Lines.Add('1. нефіксованої кількісті різних типів');

Memo1.Lines.Add('2. фіксованої кількісті тільки однакових типів');

Memo1.Lines.Add('3. розділ опису логіки програми');

Memo1.Lines.Add('4. нефіксованої кількісті однакових типів');

Memo1.Lines.Add('5. однакової кількісті різних типів');

End;

6: Begin

Label3.Caption := 'Яким ключовим словом починається розділ опису констант? ';

Memo1.Lines.Add('1. Файлова константа');

Memo1.Lines.Add('2. Const');

Memo1.Lines.Add('3. Файловий оператор');

Memo1.Lines.Add('4. Файловий масив');

Memo1.Lines.Add('5. Файлові записи');

End;

7: Begin

Label3.Caption := 'Яким ключовим словом починається заголовок модуля?';

Memo1.Lines.Add('1. Дійсного типу');

Memo1.Lines.Add('2. Unit ');

Memo1.Lines.Add('3. Типу множина');

Memo1.Lines.Add('4. Типу масив');

Memo1.Lines.Add('5. Файлового типу');

End;

8: Begin

Label3.Caption := 'Яким ключовим словом починається інтерфейсна секція?';

Memo1.Lines.Add('1. mass[1..5] of real');

Memo1.Lines.Add('2. real[1..5] of real');

Memo1.Lines.Add('3. interface');

Memo1.Lines.Add('4. readmass[1..5] of real');

Memo1.Lines.Add('5. matr[1..5] of real');

End;

9: Begin

Label3.Caption := 'Яким ключовим словом починається описова секція?';

Memo1.Lines.Add('1. 128');

Memo1.Lines.Add('2. Implementation');

Memo1.Lines.Add('3. 32767');

Memo1.Lines.Add('4. 65535');

Memo1.Lines.Add('5. 2147483647');

End;

10: Begin

Label3.Caption := 'Яким ключовим словом починається секція ініціалізації? ';

Memo1.Lines.Add('1. initialization');

Memo1.Lines.Add('2. елементи другої множини, що не містяться в першій множині');

Memo1.Lines.Add('3. елементи множини, що містяться в другій множині');

Memo1.Lines.Add('4. елементи множини, що містяться в першій множині');

Memo1.Lines.Add('5. елементи першої множини, що містяться в другій множині');

End;

11: Begin

Label3.Caption := 'Яким ключовим словом починається секція деініціалізації?';

Memo1.Lines.Add('1. одним');

Memo1.Lines.Add('2. finalization');

Memo1.Lines.Add('3. трьома');

Memo1.Lines.Add('4. чотирма');

Memo1.Lines.Add('5. п'+'"'+'ятьма');

End;

12: Begin

Label3.Caption := 'Яка схема програм реалізується у Windows ?';

Memo1.Lines.Add('1. 4 Гб');

Memo1.Lines.Add('2. Керованих подіями');

Memo1.Lines.Add('3. 512 Кб');

Memo1.Lines.Add('4. 256 Кб');

Memo1.Lines.Add('5. 64 Кб');

End;

13: Begin

Label3.Caption := 'Як розшифровується абревіатура API ?';

Memo1.Lines.Add('1. LengthSet');

Memo1.Lines.Add('2. SetString');

Memo1.Lines.Add('3. StringSet');

Memo1.Lines.Add('4. StringLength');

Memo1.Lines.Add('5. Інтерфейс прикладного програмування');

End;

14: Begin

Label3.Caption := 'Який розділ програми є обовязковим?';

Memo1.Lines.Add('1. f1 : TextFile');

Memo1.Lines.Add('2. f2 : File of integer');

Memo1.Lines.Add('3. f3 : File of real');

Memo1.Lines.Add('4. Begin..End.');

Memo1.Lines.Add('5. f5 : System.Text');

End;

15: Begin

Label3.Caption := 'Яким ключовим словом починається розділ опису типів?';

Memo1.Lines.Add('1. Low');

Memo1.Lines.Add('2. SetLow');

Memo1.Lines.Add('3. Type');

Memo1.Lines.Add('4. High');

Memo1.Lines.Add('5. LowHigh');

End;

end;

if k > 15 then

Begin

if (m>=0)and(m<4) then km := 1;

if (m>=4)and(m<7) then km := 2;

if (m>=7)and(m<9) then km := 3;

if (m>=9)and(m<12) then km := 4;

if (m>=12)and(m<=15) then km := 5;

Str(km:7,Skm);

Edit3.Text := Skm;

Exit;

End;

Sk := IntToStr(k);

Edit2.Text := Sk;

Edit1.SetFocus;

Form1.Button1.Enabled := False;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

Edit1.SetFocus;

Sk := IntToStr(k);

Edit2.Text := Sk;

Form1.Button1.Enabled := False;

end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

if not ( Key in ['1'..'5']) then

Begin

Key := #0;

Beep;

Exit;

End;

Form1.Button1.Enabled := True;

end;

end.

**Результат роботи тест-програми**

**ПРАКТИЧНА ЧАСТИНА**

**Метод Гаусса з вибором головного елементу для розв’язку СЛАР**

У лінійній алгебрі розглядаються чотири класи основних задач: розв’язок систем лінійних алгебраїчних рівнянь (СЛАР), знаходження визначників, визначення обернених матриць, знаходження особистих значень і особистих векторів матриць. Всі ці задачі мають важливе прикладне значення при розв’язанні різних проблем науки і техніки. Крім того, задачі лінійної алгебри є допоміжними при реалізації багатьох алгоритмів математики і математичної фізики.

Необхідно визначити СЛАР

а11х1 + а12 х2 + ... + а1n хn = а1,n+1 , (1.1)

а21 х1 + а22 x2 + …+ а2n хn =a2,n+1,

аn1 х1 + а n2 x2+ ...+ аnn xn = an,n+1,

де хk — невідомі величини; а ij — задані елементи розширеної матриці системи рівнянь. Для розв’язку СЛАР приміняють два класи методів: прямі та ітераційні. Прямі методи універсальні і приміняються до розв’язку систем невисокого порядку (n< 200). Ітераційні методи зручно використовувати для СЛАР високого порядку з слабо заповненими матрицями.

Метод Гаусса відноситься до прямих методів. Алгоритм методів складається з двох етапів. Перший етап називається прямим ходом і заключається у послідовному виключенні невідомих з рівнянь, починаючи з х1.

З першого рівняння системи(1.1) виражаємо невідоме х1:

х1 = (а1,n+1 - а12 х2 - ... -а1n хn )/а11, (1.2)

що можливе при а11 ≠ 0, в іншому випадку потрібно виконати перестановку рівнянь системи. Згідно формулі (1.2) потрібно кожен елемент першого рядка розширеної матриці СЛАР поділити на діагональний елемент

а1j(1) = a1j /a11. (1.3)

Потім підставляємо вираз (1.2) у всі рівняння системи, тим самим відкидаємо х1 із всіх рівнянь, крім першого. Елементи розширеної матриці перетворюємо по формулі

a i j(1) = а i j – аi 1а1j(1) ,

i = 2, 3, ..., n; j = 1, 2, ..., n+1. (1.4)

У результаті виключення першого невідомого х1 із всіх рівнянь, всі елементи першого стовпчика перетворення матриці будуть рівні нулю, крім а11(1) =1.

Невідоме х2 виражаємо із другого рівняння системи і виключаємо із останніх рівнянь і т.д. В результаті отримуємо СЛАР з верхньою трикутною матрицею, у якої всі елементи нижче головної діагоналі рівні нулю.

Запишем вирази для невідомих хk і перетворення елементів розширеної матриці системи, згідно формулам (1.2) – (1.4):

xk = (аk,n+1 -∑ аk j хj )/а kk , (1.5)

а kj (m+1)= аkj (m)/ а kk(m),

а i j (m+1)= а i j (m)- а i k(m)а kj (m).

Другий етап розв’язку СЛАР називається зворотнім ходом методу Гаусса і полягає в послідовному визначенні невідомих х k за першою формулою (1.5), починаючи з невідомого х n і закінчуючи х1 .

Точність результатів буде визначатися точністю виконання арифметичних операцій при перетворенні елементів матриці. Для зменшення похибки при діленні на діагональний елемент (друга формула (1.5)) рекомендується виконати таку перестановку рівнянь, щоб поставити на діагональ найбільший по модулю із всіх елементів цього стовпчика. Така процедура називається вибором головного елемента стовпчика. Кількість арифметичних операцій у методі Гаусса зв’язано з розмірністю системи і наближено рівно 2/3 n3 . Контроль отриманих розв’язків можна провести шляхом їх підстановки в початкову СЛАР і знаходження rk , різницю між правими і лівими частинами рівнянь:

r k = а k,n+1 - ∑ аk j х j (1.6)

При малій похибці розв’язок величини r k буде наближатися до нуля.

Ввід: 0

N – порядок системи

 Форматування 1 розширеної матриці А (N, N + 1)

Метод Гаусса 2

Вивід розв’язків X (N) 0

Мал.1.1. Блок-схема програми розв’язку СЛАР методом Гаусса

Програму розв’язку СЯАР методом Гаусса складемо з трьох блоків (мал. 1.1). В блоці О (основна програма) задамо порядок системи і звертаємось до блоку 1, в якому визначаються елементи розширеної матриці СЛАР. Потім звертаємось до блоку 2, де реалізована програма методу Гаусса. Після цього виведем на дисплей результати розв’язку СЛАР або звістку про те, що розв’язок не існує у випадку виродженої матриці (DЕТ = О).

Для загальності блок 1 оформлений у вигляді окремої підпрограми. Тут елементи розширеної матриці задаються по рядкам у діалоговому режимі з клавіатури ПЕВМ. Хоча для інших конкретних задач елементи матриці можуть визначатися шляхом обчислення за даним алгоритмом, що потребує зміни програми блоку 1. Блок 2 являє собою програму методу Гаусса, яка складається з програми прямого і зворотнього ходів. У процесі прямого ходу здійснюється перестановка рядків матриці з метою вибору головних елементів стовпців.

Особливістю програми 1.1 мовою Паскаль є введення нових типів перемінних MAT і VEC для матриці А і вектора результатів Х. Таке введення потрібне тому, що перемінні А і Х є формальними і фактичними параметрами процедур.

Для контролю програм вирахуємо систему рівнянь

х1 – 3х2 + 2х3 =7 ,

4х1 + 6х2 + х3 = 3 ,

2х1 + х2 – 2х3 = -1,

у результаті отримаємо х1 = 2, х2 = -1, х3 = 1.

**Реалізація програми для метода Гауса для СЛАР**

program Project1;

{$APPTYPE CONSOLE}

uses

 SysUtils;

Type

MAT=array [1..20,1..21] of real;

 VEK=array [1..20] of real;

Var

 A:MAT;

 X:VEK;

 I, N:integer;

 S: real;

Procedure MATR (N:integer; var A:MAT);

Var

 I, J:integer;

Begin

 for I:=1 to N do

 For J:=1 to N+1 do

 begin

 Write (' A', I:2, J:2, ' ? => ');

 readln (A[I, J])

 End

End;

Procedure Gauss (N:integer; var A:MAT; var X:VEK; var S:real);

Var

 I,J,K,L,K1,N1:integer;

 R:real;

Begin

 N1:= N+1;

 For K := 1 to N do

 begin

 K1:= K+1;

 S:= A[K,K];

 J := K;

 For I := K1 to N do

 begin

 R:= A[I,K];

 If abs (R) > abs (S) then

 begin

 S:= R;

 J:= I

 end

 End;

 If S = 0.0 then exit;

 If J <> K then

 for I:= K to N1 do

 begin

 R:= A[K, I];

 A[K, I]:= A[J, I];

 A[J, I]:= R

 end;

 For J:= K1 to N1 do A[K, J]:= A[K, J] / S;

 For I:= K1 to N do

 begin

 R:= A[I, K];

 For J:= K1 to N1 do A[I, J]:= A[I, J] - A[K, J] \*R

 End

 End;

 If S <> 0.0 then

 For I:= N downto 1 do

 begin

 S:= A[I, N1];

 For J:= I+1 to N do S:= S - A[I, J]\*X[J];

 X [I]:= S

 End

End;

begin

 Repeat

 write (' N ? => ');

 Readln (N);

 Matr (N,A);

 Gauss (N, A, X, S);

 If S <> 0.0 then

 for I:= 1 to N do

 writeln (' X', I:2, ' = ', X[I])

 Else

 writeln (' DET = 0');

 Until false ;

 Readln;

 Readln;

end.

**Результат роботи програми**

**Висновок**

При виконанні курсової роботи я поглибила свої знання в області комп’ютерної техніки, набула практичних навичок роботи з ЕОМ та інформацією представленою в ній, навчилась самостійно шукати методи розв’язку задач, освоїла прийоми систематизації знань та використала їх на практиці, глибше зрозуміла основи алгоритмізації та програмування при розв’язку задач, навчилась виконувати оформлення результатів роботи та представляти їх при захисті, освоїла основні елементи практичного програмування, уяснила роль програмного забезпечення ЕОМ, добре засвоїла принципи роботи з найбільш необхідним програмним забезпеченням та мовами програмування (Windows, MS Offise, Delphi).

**Список використаної літератури**

1. Бородич Ю.С. и др. Паскаль для персональных компьютеров. Минск, изд-во "Вышейшая школа". 1991
2. Зуев Е.А. и др. Язык прграммирования Turbo Pascal, М. "Унитех", 1992
3. Введение в Делфи 6. Электронный вариант.
4. Сухарев М.В. Основы Делфи. изд-во "Наука и техника", С-П, 2004
5. Хомоненко А. и др. Делфи 7, С-П. "БХВ-Петербург", 2006
6. Галисеев Г.В. Компоненты в Делфи 7. изд-во "Диалектика", С-П, 2004
7. Ф.В. Левитин, Алгоритмы: введение в разработку и анализ, М.”Вильямс”, 2006.
8. С.В.Поршнев, Вычислительная математика. Курс лекций, СПб,”БХВ-Петербург”, 2004.
9. К.Х. Зеленский, Компьютерные методы прикладной математики, К,”Дизайн”, 1999.
10. В.Е. Краскевич и др., Численне методы в инженерных исследованиях,-К,”Вища школа”,1986.