МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра інформатики

КУРСОВА РОБОТА

Тема:"Рішення системи лінійних рівнянь методом Крамера та за допомогою розширеної матриці"

з дисципліни "Програмування"

ПОЯСНЮВАЛЬНА ЗАПИСКА

Харків

2006

**РЕФЕРАТ**

Пояснительная записка к курсовой работе: 33 c., 1 таб., 2 рис., 5 разделов, 1 приложение, 7 источников.

Объект исследования – система линейных уравнений.

Цель работы – разработка программы, с помощью которой методом Крамера и с помощью расширенной матрицы решить систему линейных уравнений.

Метод исследования – изучение литературы, написание и отладка программ на компьютере.

Данную программу можно использовать для нахождения неизвестных системы линейных уравнений.

Разработана одна программа, которая демонстрирует нахождение неизвестных системы линейных уравнений.

Программу написано на языке С++ в среде Visual C++. Эта программа предназначена для разработки компьютерного учебного пособия по дисциплине “Программирование”. МАТРИЦА, АЛГОРИТМ, ФУНКЦИИ, ПАРАМЕТРЫ, ТИПЫ, ОПЕРАТОРЫ, ЗАГОЛОВОЧНЫЕ ФАЙЛЫ, КРАМЕР, ПЕРЕМЕННАЯ, ДЕМОНСТРАЦИОННАЯ ПРОГРАММА.

**ЗМІСТ**

Вступ

1 Постановка задачі та сфера її використання

2 Теоретична частина

2.1 Метод Крамера

3 Особливості роботи в середовищі Visual C++

4 Програмна реалізація

4.1 Опис алгоритму программи

4.2 Опис використаних програмних засобів

4.2.1 Структура програми

4.2.2 Заголовочні файли

4.2.3 Типы даних

4.2.4 Змінні

4.2.5 Ідентифікатори

4.2.6 Оператори

4.2.7 Оператори присвоювання

4.2.8 Арифметичні оператори

4.2.9 Умовні оператори

4.2.10 Оператори циклу

4.2.11 Оператори введення-виведення

4.2.12 Масивы й покажчики

4.2.13 Функції

4.3 Опис розроблених функцій

5 Інструкція користувача

Висновки

Перелік посилань

Додаток А Текст програми

**ВВЕДЕНИЕ**

В настоящее время персональный компьютер всё более и более становится неотъемлемым и необходимым предметом каждого человека. Многие из современных персональных компьютеров могут выполнить десятки миллионов операций сложения в секунду. Человеку, работающему с настольным калькулятором, потребовались бы десятилетия для того, чтобы завершить тот же самый объём вычислений, который мощный персональный компьютер выполнит за одну секунду. Сегодняшние самые быстрые суперкомпьютеры могут выполнять сотни миллиардов операций сложения в секунду – это примерно столько же, сколько сотни тысяч людей могут выполнить за год.

Стоимость аппаратных средств в последние годы существенно снизилась и достигла уровня, когда персональные компьютеры превратились в предмет массового потребления. Достижениями в отрасли информационных технологий сегодня вряд ли кого-то удивишь. Однако информационные технологии сегодня играют слишком важную роль в нашей жизни, а программирование – двигатель развития информационных технологий.

Язык С++ развился из С, который в свою очередь был создан на основе двух предшествующих языков – BCPL и B. Язык BCPL был создан в 1967 году Мартином Ричардом как язык для написания компиляторов и программного обеспечения операционных систем. Кен Томпсон предусмотрел много возможностей в своем языке В – дубликате BCPL и использовал В для создания ранних версий операционной системы UNIX.

Язык С был развит из В Деннисом Ритчи в Bell Laboratories. С использует многие важные концепции BCPL и В, а также добавляет типы данных и другие свойства. Первоначально С приобрел широкую известность как язык разработки операционной системы UNIX. Сегодня фактически все новые операционные системы написаны на С или С++. В течение двух последних десятилетий С стал доступным для большинства компьютеров. С независим от аппаратных средств. При тщательной разработке на С можно написать мобильные программы, переносимые на большинство компьютеров.

 С++ - расширение С – был разработан Бьерном Строустропом в начале 80-х годов в Bell Laboratories. С++ обеспечивает возможность объектно-ориентированного программирования.

С++ - это гибридный язык, он предоставляет возможность программировать и в стиле С, и в объектно-ориентированном стиле, и в обоих стилях сразу. Его синтаксис, стиль и основные принципы стали стандартом при разработке новых языков. Благодаря своей универсальности он все чаще используется при описании алгоритмов и технологий программирования. Особенность языка С заключается в том, что он позволяет непосредственно манипулировать битами, байтами, машинными словами и указателями. Это делает его очень удобным для системного программирования, в котором эти операции широко распространены.

Язык С был создан для программистов, учитывая их интересы, и многократно проверялся на практике, прежде чем был окончательно реализован. Именно поэтому языки С и С++ стали наиболее популярными среди программистов высокого уровня [4].

В данной курсовой работе я представил программу, написанную на языке С++, с помощью которой мы можем решить систему линейных уравнений.

**1 ПОСТАНОВКА ЗАДАЧИ И СФЕРА ЕЁ ИСПОЛЬЗОВАНИЯ**

Целью данной задачи является решение системы линейных уравнений, т.е. нахождение её неизвестных х1, х2, х3 методом Крамера и с помощью расширенной матрицы.

Начальными данными являются коэффициенты при неизвестных х1, х2, х3 и свободные члены. Свободные члены и коэффициенты при неизвестных являются главными данными, которые программа записывает в свою память.

Решение систем линейных уравнений является одной из важ­ных вычислительных задач. Большинство задач вычислительной прак­тики сводятся к решению систем линейных уравнений. Это задачи из области электротехники, радиоэлектроники, механики, статистики. Серьез­ные практические задачи часто приводят к таким системам, кото­рые содержат сотни и даже тысячи линейных уравнений. Без помощи компьютера, эти системы решить невозможно.

Данная задача используется в математической сфере. Целесообразность решения задачи автоматизированным способом позволяет сократить время, затраченное на её решение по сравнению с ручным способом, снижается вероятность допущения ошибок, повышается точность полученных результатов

**2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**2.1 Метод Крамера**

Для того, чтобы понять суть метода Крамера, необходимо знать такие понятия, как определитель и матрица.

Так как в нашем случае используется определитель 3-го порядка, то введём определение определителя 3-го порядка.

Определителем 3-го порядка называется число, обозначаемое

 a11 a12 a13

 a21 a22 a23

 a31 a32 a33

и равное a11a22a33+a13a22a32+a12a23a31-a13a22a31-a12a21a33-a11a23a32.

Определитель обозначается Δ.

Матрицей называется таблица, содержащая m строк и n столбцов вида

 a11 a12 ..... a1n

 a21 a22 ..... a11

 . . . . . .

 am1 am2 ....amn

где aij – элемент матрицы, находящийся в i-той строке и j-ом столбце, где i изменяется от 1 до m, j – от 1 до n, m и n – порядки матрицы. Отсюда вытекает определение системы линейных уравнений.

Система вида a11x1+a12x2+.....+a1nxn=b1

 a21x1+a22x2+.....+a2nxn=b2

 . . . . . . . . . . .

 am1x1+am2x2+.....+amnxn=bm

где aij (i изменяется от 1 до m; j – от 1 до n) – коэффициенты при неизвестных x1, x2,…., xn, bi (i изменяется от 1 до m) – свободные члены, называется системой линейных уравнений.

Теперь непосредственно перейдём к методу Крамера.

Рассмотрим систему вида

 a11x1+a12x2+.....+a1nxn=b1

 a21x1+a22x2+.....+a2nxn=b2

 . . . . . . . . . . .

 am1x1+am2x2+.....+amnxn=bm

Назовём данную систему “система 1”.

Умножим 1-е уравнение системы 1 на А11. Умножим 2-е уравнение системы 1 на А21. Умножим n-е уравнение системы 1 на Аn1. Сложим полученные уравнения:

(a11А11+a21А21+….+an1Аn1)x1+(a12А12+a22А21+....+an1Аn1)x1+....+(a1nА11+a2nА21+ +....+annАn1)xn=b1А11+b2А21+....+bnАn1

detA x1=detA1,

где A1 – матрица, полученная из матрицы А заменой 1-го столбца столбцом свободных членов.

detA x2=detA2,

где A2 – матрица, полученная из матрицы А заменой 2-го столбца столбцом свободных членов.

detA xn=detAn,

где An – матрица, полученная из матрицы А заменой n-го столбца столбцом свободных членов.

Если обозначим Δ=detA, то получим

 Δ x1=Δ1

 Δ x2=Δ2

 . . . .

 Δ xn=Δn

где Δi – определитель, полученный из определителя Δ заменой i-го столбца столбцом свободных членов системы 1.

Назовём данную систему “система 2”.

Рассмотрим 3 случая:

1) Если в системе 2 Δ≠0, то x1=Δ1/Δ, x2=Δ2/Δ, ….. , xn=Δn/Δ. Полученные формулы называются формулами Крамера.

2) Если в системе 2 Δ=0 и все Δi=0 для любого i, изменяющегося от 1 до n, то система имеет бесчисленное множество решений.

3)Если в системе 2 Δ=0 и существует Δi≠0, то решения системы 2 не существуют [1].

**3 ОСОБЕННОСТИ РАБОТЫ В СРЕДЕ Visual C++**

**Язык программирования Visual C++** в настоящее время является **самым популярным** языком в мире. Программу сколь угодно большой сложности Вы можете написать на любом языке программирования, но если требуется, чтобы она работала быстро и надёжно, вы вынуждены писать её на Visual C++.

Популярность этого языка обусловлена ещё и мощными визуальными средствами разработки интерфейса пользователя. Высокая производительность работы – ещё одно преимущество Visual C++.

В среде Visual C++ можно строить различные типы проектов. Такие проекты после их создания можно компилировать и запускать на исполнение. Фирма Microsoft разработала специальный инструментарий, облегчающий и ускоряющий создание проектов в среде Visual C++.

Рассмотрим некоторые типы проектов, которые можно создавать при помощи различных средств (мастеров проектов) Microsoft Visual C++:

MFC AppWizard (exe) – при помощи мастера приложений можно создать проект Windows-приложения которое имеет однодокументный, многодокументный или диалоговый интерфейс. Однодокументное приложеие может предоставлять пользователю в любой момент времени работать только с одним файлом. Многодокументное приложение, напротив, может одновременно представлять несколько документов, каждый в собственном окне. Пользовательский интерфейс диалогового приложения представляет собой единственное диалоговое окно.

MFC AppWizard (dll) – этот мастер приложений позволяет создать структуру DLL, основанную на MFC. При помощи него можно определить характеристики будующей DLL.

AppWizard ATL COM – это средство позволяет создать элемент управления ActiveX или сервер автоматизации, используя новую библиотеку шаблонов ActiveX (ActiveX Template Library - ATL). Опции этого мастера дают возможность выбрать активный сервер (DLL) или исполняемый внешний сервер (exe-файл).

Custom AppWizard – при помощи этого средства можно создать пользовательские мастера AppWizard. Пользовательский мастер может базироваться на стандартных мастерах для приложений MFC или DLL, а также на существующих проектах или содержать только определеямые разработчиком шаги.

DevStudio Add-in Wizard – мастер дополнений позволяет создавать дополнения к Visual Studio. Библиотека DLL расширений может поддерживать панели инструментов и реагировать на события Visual Studio.

MFC ActiveX ControlWizard - мастер элементов управления реализует процесс создания проекта, содержащего один или несколько элементов управления ActiveX, основанных на элементах управления MFC.

Win32 Application – этот мастер позволяет создать проект обычного Window-приложения. Проект создается незаполненным, файлы с исходным кодом в него следует добавлять вручную.

Win32 Console Application – мастер создания проекта консольного приложения. Консольная приложение – это программа, которая выполняется из командной cтроки окна DOS или Windows и не имеет графического интерфейса (окон). Проект консольного приложения создается пустым, предполагая добавление файлов исходного текста в него вручную.

Win32 Dynamic-Link Library – создание пустого проекта динамически подключаемой библиотеки. Установки компилятора и компоновщика будут настроены на создание DLL. Исходные файлы следует добавлять вручную.

Win32 Static Library – это средство создает пустой проект, предназначенный для генерации статической (объектной) библиотеки. Файлы с исходным кодом в него следует добавлять вручную [7].

Исходя из выше перечисленных функций языка Visual C++ можно с уверенностью сказать, что этот язык является сверхнадёжным и обладает обширными функциями, благодаря которым можно написать программы различной сложности.

**4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

**4.1 Описание алгоритма программы**

Разработанная в ходе выполнения курсовой работы программа, состоит из двух функций. Алгоритм выполнения программы изображен на рисунке 4.1 [3].

Ввод матрицы Аij

 начало

Ввод матрицы Bij

Kramer (A,B)

конец

Рисунок 4.1 – Алгоритм выполнения программы

**4.2 Описание использованных програмних средств**

Любая программа представляет собой последовательность инструкций в машинных кодах, которые управляют поведением определенного вычислительного устройства [4].

**4.2.1 Структура программы**

Все программы на языке С++ состоят из одной или нескольких функций. В любом случае программа должна содержать функцию main(), которая при выполнении программы вызывается первой. Определение функции main состоит из заголовка void main(), и последовательности инструкций, заключённых в фигурные скобки. Слово void в заголовке говорит о том, что функция не должна вырабатывать и возвращать значение. Все программы должны иметь функцию с именем main.

Общий вид программы на языке С++ показан в следующем примере. Функции с именами fun1(),…,funn() определяются пользователем [2]. Пример:

подключение заголовочных файлов

объявление глобальных переменных

тип\_возвращаемого\_значения main(список\_параметров)

{

 последовательность операторов

}

тип\_возвращаемого\_значения fun1(список\_параметров)

{

 последовательность операторов

}

.

.

.

тип\_возвращаемого\_значения funn(список\_параметров)

{

 последовательность операторов

}

**4.2.2 Заголовочные файлы**

Любая программа на языке С++ содержит подключение заголовочных файлов с помощью директивы препроцессора #include<имя\_заголовочного\_файла.h>.

Директива #include вынуждает компилятор считать и подставить в исходный файл с заданным именем. Это имя заключается в двойные или угловые скобки.

Кавычки и угловые скобки, в которых указываются имена включаемых файлов, определяют способ их поиска на жестком диске. Если имя файла содержится в угловых скобках, он должен находиться в каталоге, указанном компилятором. Если имя файла заключено в кавычки, как правило, его поиск выполняется в рабочем каталоге. Если файл не найден, то поиск повторяется так, будто имя файла содержалось в угловых скобках.

Например:

#include <iostream.h>

#include <conio.h>

#include <iomanip.h>

В первом примере с помощью директивы #include подключают заголовок ввода-вывода <iostream.h>. Этот файл содержит информацию, необходимую, чтобы компилировать программу, которая использует cin и cout.

Во втором примере директивы #include подключают заголовок <iomanip.h>, который даёт указание препроцессору С++ включить в программу параметризованный манипулятор потока setw. Он предназначен для манипулирования выходными величинами на экране, например:

cout << setw(3)<<year

Обращение setw(3) определяет, что следующая выходная величина будет напечатана с шириной (размером) поля 3, т.е. её значение будет содержать по крайней мере 3 символьных позиции.

В третьем примере с помощью директивы #include подключают заголовок <conio.h>, который содержит функцию задержки экрана getch() [4].

**4.2.3 Типы данных**

В языке С++ существуют 5 типов данных: символ, целое число, число с плавающей запятой, число с плавающей запятой удвоенной точности и переменная, не имеющая значение. Им соответствуют следующие ключевые слова: char, int, float, double, void. Все другие типы данных в языке С++ создаются на основе элементарных типов, указанных выше. Размер переменных и диапазон их значений зависит от типа процессора и компилятора.

К пяти основным типам данных, определенных в языке С, язык С++ добавляет еще такой тип как bool.

Тип void используется для определения функции, не возвращающей никаких значений, либо для создания обобщенного указателя.

В таблице 1.1 приведен перечень базовых типов переменных и их размер [2,5].

Таблица 1.1

Базовые типы данных для ПК на базе платформы Іntel

|  |  |  |
| --- | --- | --- |
| Тип | Размер, байт | Значения |
| Bool | 1 | true (1) или false(0) |
| unsіgned short іnt | 2 | от 0 до 65 535 |
| short іnt | 2 | от -32 768 до 32 767 |
| unsіgned long іnt | 4 | от 0 до 4 294 967 295 |
| long іnt | 4 | от -2 147 483 648 до 2 147 483 647 |
| іnt (16 разрядов) | 2 | от -32 768 до 32 767 |
| іnt (32 розряда) | 4 | от -2 147 483 648 до 2 147 483 647 |
| unsіgned іnt (16 разрядов) | 2 | от 0 до 65 535 |

|  |  |  |
| --- | --- | --- |
| unsіgned іnt (32 розряда) | 4 | от 0 до 4 294 967 295 |
| Char | 1 | от 0 до 256 |
| Float | 4 | от 1.2е-38 к 3.4е38 |
| Double | 8 | от 2.2е-308 к 1.8е308 |
| long double | 10 | от 3.4е-4932 к 3.4е 4932 |
| Voіd | 2 или 4 | - |

**4.2.4 Переменные**

Переменная – это имя ячейки памяти, которую можно использовать для хранения модифицируемого значения. Все переменные должны быть объявлены до своего использования.

Ниже приведен общий вид объявления переменной.

тип список\_переменных;

Например:

int i,j;

При объявлении переменной ей можно присвоить начальное значение. Общий вид инициализации выглядит следующим образом:

тип имя\_переменной = значение;

Например:

int a=0;

Глобальные и локальные статические переменные инициализируются только при запуске программы. Локальные переменные инициализируются каждый раз при входе в блок, где они описаны. Неинициализированные локальные переменные имеют неопределенное значение, пока к ним не будет применен оператор присваивания. Неинициализированные глобальные переменные и локальные статические переменные автоматически устанавливаются равными нулю [5].

Практически все символы можно вывести на печать, заключив их в одиночные кавычки. Однако некоторые символы, например, символ перехода на новую строку, невозможно ввести в строку с клавиатуры. Для этого в языке С/С+ + предусмотрены специальные управляющие символьные константы.

**4.2.5 Идентификаторы**

В языке С/С++ имена переменных, функций, меток и других объектов, определенных пользователем, называются идентификаторами. Идентификаторы могут состоять из одного или нескольких символов. Первый символ идентификатора должен быть буквой или символом подчеркивания, а следующие символы должны быть буквами, цифрами или символами подчеркивания.

В языке С++ нет ограничений на длину идентификаторов, и значащими считаются, по крайней мере, 1024 первых символа.

Символы, набранные в верхнем и нижнем регистре, различаются т. к. язык С/С++ чувствителен к регистру. Следовательно, imin Imin, IMIN – это различные идентификаторы. Переменная с именем, например, а1 в языке С++ записывается как а1. В противном случае компилятор выдаст сообщение об ошибке.

Ключевые слова нельзя использовать в качестве идентификаторов и они не должны совпадать с именами функций из стандартных библиотек [2].

**4.2.6 Операторы**

В языке С/С++ предусмотрено большое количество операторов. В этих языках операторам уделяется гораздо больше внимания, чем в большинстве других языков. Операторы разделяются на 4 основные группы: арифметические, сравнения, логические и побитовые.

**4.2.7 Оператор присваивания**

Оператор присваивания можно использовать в любом корректном выражении. Общий вид оператора присваивания выглядит следующим образом:

имя\_переменной = выражение

Здесь выражение может состоять как из отдельной константы, так и комбинации сложных операторов [4].

**4.2.8 Арифметические операторы**

Операторы +, -, \* и / выполняются точно так же, как и в большинстве других языков программирования. Их можно применять практически к любым встроенным типам данных. Если оператор / применяется к целому числу или символу дробная часть отбрасывается.

Оператор %, как и в других языках, возвращает остаток целочисленного деления. Однако он не применим к числам с плавающей точкой.

Унарный минус умножает число на -1.

Оператор ++ добавляет 1 к своему операнду, а оператор – – вычитает ее.

Операторы сравнения и логические операторы имеют более низкий приоритет, чем арифметические операции.

**4.2.9 Условные операторы**

Оператор условного перехода имеет следующий формат записи:

іf (А) оператор 1 ;

 еlse оператор 2 ;

где А - выражение. Если значение этого выражения "истина" (не нуль), то выполняется оператор1, если же оно "ложь", то выполняется оператор2; в случае, когда выражение ошибочное и отсутствует область else - выполняется следующий оператор [4]. Например:

 if (a=1) b=a\*2;

 else b=0;

**4.2.10 Операторы цикла**

В языке С/С++ операторы цикла предназначены для выполнения повторяющихся инструкций, пока действует определенное правило. Это условие может быть как задано заранее(в цикле for), так и меняться во время выполнения цикла(в операторах while и\_do-while)

В программе широко используется цикл for.

Оператор цикла for имеет следующую форму записи:

for ( [выражение1;] [выражение2;] [выражение3] ) оператор;

где выражение1 - выражение инициализации, которое обычно используется для установки начального значения; это выражение присваивания (необязательный параметр); выражение2 - выражение условия, которое определяет, при каком условии цикл будет повторяться (необязательный параметр); выражение3 - выражение итерации, которая определяет шаг изменения сменных, которые руководят циклом после каждого выполнения (необязательный параметр) [2].

Этот оператор реализуется таким образом:

- сначала выполняется выражение инициализации;

- вычисляется условное выражение;

- если результат условного выражения "истина" (не равняется нулю), то выполняется оператор цикла;

- вычисляется выражение итерации;

- снова проверяется условие;

- как только условное выражение становится равным нулю ("неправда"), управления передается оператору, который располагается за оператором цикла for.

Поскольку проверка условия производится перед циклом, то цикл может ни разу не выполняться, если условие сразу будет ошибочным.

Пример использования цикла for:

for(i=0;i<3;i++)

for(j=0;j<3;j++)

cin>>A[i][j];

**4.2.11 Операторы ввода-вывода**

При вводе-выводе потока все данные рассматриваются как поток отдельных байтов.

Чтобы использовать функции ввода-вывода необходимо подключить библиотеку ввода-вывода

#include <iostream.h>

Формат записи операторов ввода-вывода:

cin [ >> values ];

cout << data [ << data << "\n"] [5];

Например:

cin>>А[i][j];

cout<<"x1="<<x1<<endl;

**4.2.12 Массивы и указатели**

Массив – это последовательная группа ячеек памяти, которые имеют одинаковый тип и имеют общее имя. Доступ к отдельному элементу массива осуществляется с помощью индекса.

В языке С/С++ предусмотрены многомерные массивы. Простейшим из них является двумерный. Объявление двумерного массива А, состоящего из 3 строк и 3 столбцов выглядит следующим образом:

double А[3][3];

Обращение к элементу двумерного массива выглядит так:

А[0][0], А[0][1] и т.д.

В языке С/С++ указатели и массивы тесно связаны между собой. Как известно, имя массива без индекса – это указатель на его первый элемент [4]. Рассмотрим, например, следующий массив:

double А[3][3];

Следующие два выражения абсолютно идентичны:

А

&А[0][0]

Доступ к элементам массива может осуществляться следующим образом:

А[i][j];

\*(\*(А+i)+j);

Записи являются идентичными.

**4.2.13 Функции**

Функция – это процедура(подпрограмма), которая несёт законченную смысловую нагрузку.

Определение функции состоит из заголовка функции и тела функции, которое заключается в ( ) и несёт смысловую нагрузку.

Общий вид функции выглядит следующим образом:

тип\_функции имя\_функции(список\_аргументов)

{ тело\_функции };

Тип\_функции – это тип значения, которое возвращает функция. Если функция не возвращает никакого значения, то ей тип всегда void. Возврат значения из функции происходит с использованием оператора return, который имеет следующий вид

return выражение

Операторов return может быть сколько угодно, но функция возвращает всегда только одно значение.

По умолчанию если тип функции не определён явно, то функция имеет тип int.

Список\_аргументов может быть либо пустим, либо, например

float fun(int, float);

Тело\_функции – это всегда блок или составной оператор, т.е. последовательность описаний и операторов, заключённых в { } [6].

Для того, чтобы функция выполнила некоторые действия, она должна быть вызвана. Вызов функции имеет вид

имя\_функции(список\_фактических\_параметров);

Например:

Kramer(A,B);

**4.3 Описание разработанных функций**

Разработанная в ходе выполнения работы программа содержит функцию Kramer.

Функция Kramer решает систему уравнений, заданную матрицей коэффициентов при неизвестных и столбцом свободных членов по методу Крамера, и выводит полученные значения неизвестных на экран.

Блок-схема выполнения функции изображена на рисунке 4.3.1 [3].

начало

det, det1, det2, det3

x1 = det1/det;

x2 = det2/det;

x3 = det3/det;

вывод

х1,х2,х3

конец

Рисунок 4.3.1 – Алгоритм функции Kramer.

Функция Kramer имеет тип возвращаемого значения void, т. к. она ничего не возвращает, а только выводит найденные неизвестные.

**5 ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ**

Программа, которая была разработана в ходе выполнения курсовой работы, позволяет с помощью метода Крамера и с помощью расширенной матрицы решить систему линейных уравнений.

Минимальные системные требования для работы программы:

- OS MS-DOS или Windows

- 5 МБ на жестком диске

- 8 МБ оперативной памяти

Прилагаемая к курсовой работе дискета содержит следующие файлы:

- Kramer.exe (разработанная программа)

- Kramer.cpp (текст программы)

- Курсовой.doc (пояснительная записка)

- Данные.txt (исходные данные)

- Результаты. txt (полученные результаты)

- Readme. txt (инструкция пользователя)

Для запуска программы Kramer необходимо запустить находящуюся на дискете программу Kramer.exe.

В ответ на запрос программы: «"Vvedite koefisienti pri neizvestnih x1,x2,x3 v 1-om,2-om,3-em yravneniax"» необходимо ввести матрицу коэффициентов при неизвестных. Числа вводятся через пробел или Enter. Для ввода десятичных чисел целая часть отделяется от дробной точкой. После ввода всех девяти чисел (в случае ввода их через пробел) для дальнейшей работы программы необходимо нажать клавишу Enter. В ответ на запрос программы: "Vvedite cvobodnii chleni v 1-om,2-om,3-em yravneniax"» необходимо ввести матрицу-столбец свободных членов. После ввода всех 3 чисел (в случае ввода их через пробел) для дальнейшей работы программы необходимо нажать клавишу Enter. Полученное решение системы уравнений методом Крамера включает в себя систему линейных уравнений, расширенную матрицу, результаты нахождения определителя матриц и конечные решения.

**ВЫВОДЫ**

В ходе выполнения курсовой работы была разработана программа, с помощью которой можно решить систему линейных уравнений по методу Крамера и с помощью расширенной матрицы.

Ниже приведен результат работы программы

Vvedite koefisienti pri neizvestnih x1,x2,x3 v 1-om,2-om,3-em yravneniax

3 6 1 2 1 0 1 5 -1

Vvedite cvobodnii chleni v 1-om,2-om,3-em yravneniax

20 0 9

Dannaya sistema vigladit tak

3x1+6x2+1x3=20

2x1+1x2+0x3=0

1x1+5x2-1x1=9

C pomochy rashirenoi matrici mi mogem predstaviy dannyy sistemy tak:

3 6 1 |20

2 1 0 |0

1 5 -1|9

Reshenie sistemi yravnenii metodom Kramera

Nahogdenie opredelitelei

opredelitel=18

opredelitel1=-29

opredelitel2=58

opredelitel3=99

x1=-1.61111

x2=3.22222

x3=5.5

**СПИСОК ЛИТЕРАТУРЫ**

1. Апатёнок Р.Ф. Элементы линейной алгебры и аналитической геометрии. – Минск: Вышэйшая школа, 1977 г. – 257 с.
2. Глушаков С.В. и др. Язык программирования С++. —Харьков: Фолио, 2002. — 500 с.
3. ГОСТ 19.701—90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.— М.: 1991.
4. Дейтел Х.М. Дейтел П.Дж. Как программировать на С++: Пер. с англ. – М.: ЗАО “Издательство БИНОМ”, 2000 г. – 1024 с.: ил.
5. Страуструп Б. Язык программирования С++. Часть 1. — Киев: "ДиаСофт", 1993. — 264 с.
6. Страуструп Б. Язык программирования С++. Часть 2. — Киев: "ДиаСофт", 1993. — 296 с.
7. Холзнер С. VISUAL C++6: учебный курс — СПб: Издательство “ Питер”, 2000. —576 с.

**Приложение А**

Текст программы

#include <iostream.h>

#include <iomanip.h>

#include <conio.h>

void Kramer(double C[3][3],double D[3][1]);//Prototip funkcii

void main()

{

int i,j;

double A[3][3],B[3][1];

char z='y';

cout<<"\n"<<endl;

cout<<"\n"<<endl;

cout<<"Vipolnil stydent gryppi KI-05-3"<<endl;

cout<<"Homenko Igor"<<endl;

cout<<"\n"<<endl;

cout<<"\n"<<endl;

while (z=='y')

{

cout<<"Vvedite koefisienti pri neizvestnih x1,x2,x3 v 1-om,2-om,3-em yravneniax"<<endl;

for(i=0;i<3;i++)//vvod matrici koeficientov pri neizvesnix

for(j=0;j<3;j++)

cin>>A[i][j];

cout<<"\n"<<endl;

cout<<"Vvedite cvobodnii chleni v 1-om,2-om,3-em yravneniax"<<endl;

for (i=0;i<3;i++)// Vvod stolbca svobodnix chlenov

cin>>B[i][0];

cout<<"\n"<<endl;

cout<<"Dannaya sistema vigladit tak"<<endl;//Vvivod na ekran SLY

if(A[0][1]>=0 && A[0][2]>=0)

cout<<A[0][0]<<"x1"<<"+"<<A[0][1]<<"x2"<<"+"<<A[0][2]<<"x3"<<"="<< B[0][0]<<endl;

if(A[0][1]>=0 && A[0][2]<0)

cout<<A[0][0]<<"x1"<<"+"<<A[0][1]<<"x2"<<A[0][2]<<"x3"<<"="<<B[0][0] <<endl;

if(A[0][2]>=0 && A[0][1]<0)

cout<<A[0][0]<<"x1"<<A[0][1]<<"x2"<<"+"<<A[0][2]<<"x3"<<"="<<B[0][0] <<endl;

if(A[0][1]<0 && A[0][2]<0)

cout<<A[0][0]<<"x1"<<A[0][1]<<"x2"<<A[0][2]<<"x3"<<"="<<B[0][0] <<endl;

if(A[1][1]>=0 && A[1][2]>=0)

cout<<A[1][0]<<"x1"<<"+"<<A[1][1]<<"x2"<<"+"<<A[1][2]<<"x3"<<"="<< B[1][0]<<endl;

if(A[1][1]>=0 && A[1][2]<0)

cout<<A[1][0]<<"x1"<<"+"<<A[1][1]<<"x2"<<A[1][2]<<"x3"<<"="<<B[1][0] <<endl;

if(A[1][2]>=0 && A[1][1]<0)

cout<<A[1][0]<<"x1"<<A[1][1]<<"x2"<<"+"<<A[1][2]<<"x3"<<"="<<B[1][0] <<endl;

if(A[1][1]<0 && A[1][2]<0)

cout<<A[1][0]<<"x1"<<A[1][1]<<"x2"<<A[1][2]<<"x3"<<"="<<B[1][0]

<<endl;

if(A[2][1]>=0 && A[2][2]>=0)

cout<<A[2][0]<<"x1"<<"+"<<A[2][1]<<"x2"<<"+"<<A[2][2]<<"x3"<<"="<< B[2][0]<<endl;

if(A[2][1]>=0 && A[2][2]<0)

cout<<A[2][0]<<"x1"<<"+"<<A[2][1]<<"x2"<<A[2][2]<<"x3"<<"="<<B[2][0] <<endl;

if(A[2][2]>=0 && A[2][1]<0)

cout<<A[2][0]<<"x1"<<A[2][1]<<"x2"<<"+"<<A[2][2]<<"x3"<<"="<<B[2][0] <<endl;

if(A[2][1]<0 && A[2][2]<0)

cout<<A[2][0]<<"x1"<<A[2][1]<<"x2"<<A[2][2]<<"x3"<<"="<<B[2][0]

<<endl;

cout<<"\n"<<endl;

cout<<"C pomochy rashirenoi matrici mi mogem predstaviy dannyy sistemy tak:"<<endl;//Vivod na ekran rachirennoi matrici

cout<<A[0][0]<<setw(4)<<A[0][1]<<setw(4)<<A[0][2]<<setw(5)<<"|"<<setw (5)<<B[0][0]<<endl;

cout<<A[1][0]<<setw(4)<<A[1][1]<<setw(4)<<A[1][2]<<setw(5)<<"|"<<setw (5)<<B[1][0]<<endl;

cout<<A[2][0]<<setw(4)<<A[2][1]<<setw(4)<<A[2][2]<<setw(5)<<"|"<<setw (5)<<B[2][0]<<endl;

cout<<"\n"<<endl;

Kramer(A,B);//vizov funkcii Kramer

cout<<"Dlya novogo vvoda nagmite 'y', dla vihoda-'n' \n";

cin>>z;

}

}

void Kramer(double C[3][3],double D[3][1])//realizacia funkcii Kramer

{

double det,det1,det2,det3,x1,x2,x3;

cout<<"Nahogdenie opredelitelei"<<endl;

det=C[0][0]\*C[1][1]\*C[2][2]+C[1][0]\*C[2][1]\*C[0][2]+C[0][1]\*C[1][2]\*C[2][0]-C[0][2]\*C[1][1]\*C[2][0]-C[1][0]\*C[0][1]\*C[2][2]-C[0][0]\*C[2][1]\*C[1][2];

cout<<"opredelitel="<<det<<endl;//vichislenie opredelitela

det1=D[0][0]\*C[1][1]\*C[2][2]+D[1][0]\*C[2][1]\*C[0][2]+C[0][1]\*C[1][2]\*D[2][0]-C[0][2]\*C[1][1]\*D[2][0]-D[1][0]\*C[0][1]\*C[2][2]-D[0][0]\*C[2][1]\*C[1][2];

cout<<"opredelitel1="<<det1<<endl;//vichislenie opredelitela1

det2=C[0][0]\*D[1][0]\*C[2][2]+C[1][0]\*D[2][0]\*C[0][2]+D[0][0]\*C[1][2]\*C[2][0]-C[0][2]\*D[1][0]\*C[2][0]-C[1][0]\*D[0][0]\*C[2][2]-C[0][0]\*D[2][0]\*C[1][2];

cout<<"opredelitel2="<<det2<<endl;//vichislenie opredelitela2

det3=C[0][0]\*C[1][1]\*D[2][0]+C[1][0]\*C[2][1]\*D[0][0]+C[0][1]\*D[1][0]\*C[2][0]-D[0][0]\*C[1][1]\*C[2][0]-C[1][0]\*C[0][1]\*D[2][0]-C[0][0]\*C[2][1]\*D[1][0];

cout<<"opredelitel3="<<det3<<endl;//vichislenie opredelitela3

cout<<"\n"<<endl;

if(det!=0)

 {x1=det1/det;//vichislenie x1

x2=det2/det;//vichislenie x2

x3=det3/det;//vichislenie x3

cout<<"x1="<<x1<<endl;

cout<<"x2="<<x2<<endl;

cout<<"x3="<<x3<<endl;}

 else

cout<<"Sistema ne imeet reshenii, tak kak opredelitel=0"<<endl;

getch() ;

}

**ЛИСТ-ЗАДАНИЕ**

Методом Крамера и с помощью расширенной матрицы решить систему линейных уравнений, например,

3x1+6x2+x3=20

2x1+x2=0

 x1+5x2-x3=9