МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

М Е Т О Д И Ч Н І В К А З І В К И

до виконання лабораторних робіт з дисципліни

"Інформатика, програмування та числові методи"

для студентів напряму підготовки

6.050402 – Ливарне виробництво

усіх форм навчання

Київ
Політехніка
2015

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Методичні вказівки
до виконання лабораторних робіт з дисципліни
"Інформатика, програмування та числові методи"
для студентів напряму підготовки
6.050402 – Ливарне виробництво
усіх форм навчання

Затверджено
на засіданні кафедри
ливарного виробництва
чорних і кольорових
металів, протокол № 5
від 12 листопада 2014 р.

Київ
Політехніка
2015

Методичні вказівки до виконання лабораторних робіт з дисципліни "Інформатика, програмування та числові методи" для студентів напряму підготовки „Ливарне виробництво” / Укладачі В.А. Клименко, О.М. Доній В.П. Самарай – К.: „ВПК «Політехніка»”, 2015. – 80 с.

Гриф надано Вченою радою ІФФ НТУУ «КПІ»
(Протокол № ___/14 від _____ 2014 р.)

Методичні вказівки

до виконання лабораторних робіт з дисципліни

"Інформатика, програмування та числові методи"

для студентів напряму підготовки

6.050402 – Ливарне виробництво

усіх форм навчання

Укладачі: Доній Олександр Миколайович, канд. техн. наук, доцент
Самарай Валерій Петрович, канд. техн. наук, доцент
Клименко Віктор Анатолійович, асистент

Відповідальний
редактор: Г.Є. Федоров, кандидат технічних наук, доцент

Рецензент: В.М Рибак, кандидат технічних наук, доцент

За редакцією укладачів

Надруковано з оригінал-макета замовника

Темплан 2015 р.

ЗМІСТ

ВСТУП

Методичні вказівки вміщують інформацію із проведення лабораторних робіт з дисципліни "Інформатика, програмування та числові методи" для студентів, які навчаються за напрямом підготовки «Ливарне виробництво». У зв’язку з необхідністю приділення підвищеної уваги до самостійної роботи студентів, методичні вказівки супроводжуються теоретичними поясненнями, відомостями про принципи будови та застосування загальних елементів

Матеріал викладено з урахуванням найбільшої практичної спрямованості, включаючи питання використання математичного оброблення результатів безпосередньо у галузі ливарного виробництва.

У кожній лабораторній роботі наведені мета, основні теоретичні положення, методики проведення і розрахунку, контрольні запитання та індивідуальні завдання.

Виконання кожної лабораторної роботи передбачає самостійне ознайомлення з теорією процесу програмування; набуття вміння створення алгоритмів, оброблювати експериментальні дані і за отриманими результатами зробити правильні висновки.

До початку лабораторного заняття студент має підготувати протокол виконання лабораторної роботи.

ЛАБОРАТОРНАЯ РАБОТА №1

Язык программирования VBA. Основные понятия и область применения.

Цель: ознакомится с назначением, возможностями и областями применения VBA.

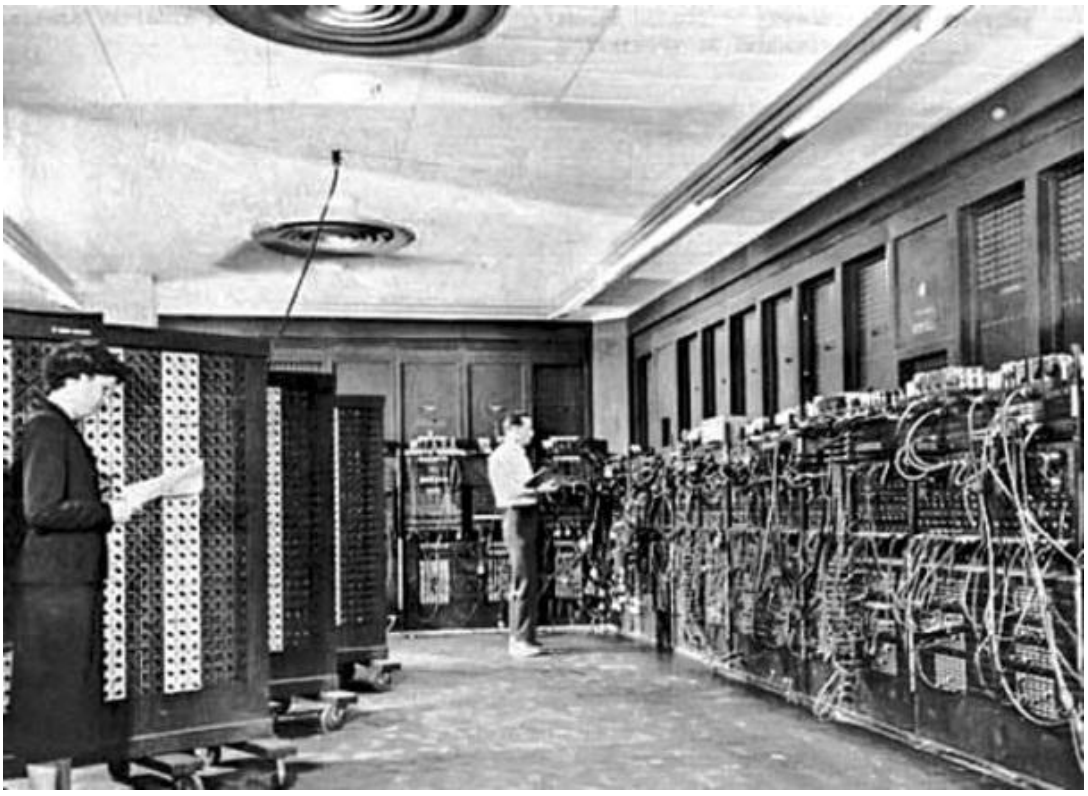
Основные вопросы.

1. История развития вычислительной техники.
2. Что такое язык программирования и компьютерные программы.
3. Возникновение и развитие алгоритмического языка Basic
4. Преимущества и недостатки область применения VBA

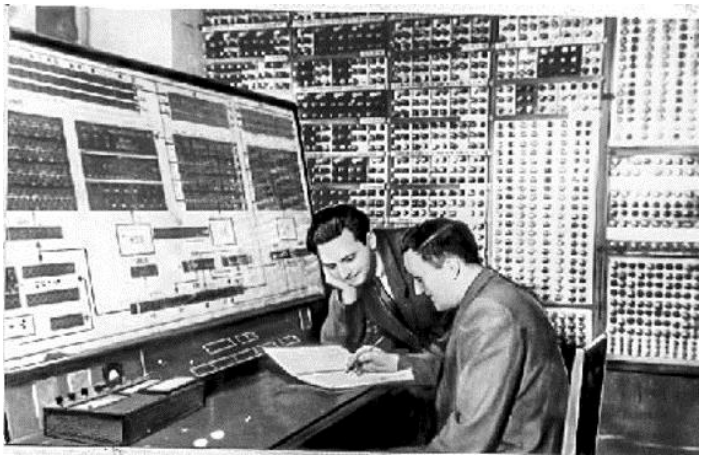
История развития вычислительной техники.

Персональный компьютер (ПК) – величайшее из изобретений XX века. Никакое другое устройство не оказало столь большого влияния на наш образ жизни, на то, как мы работаем и как общаемся.

Если проследить родословную ПК, корнем фамильного древа окажется Electronic Numerical Integrator and Computer (электронный цифровой интегратор и вычислитель), или, короче, ENIAC. Машина массой в три тонны, разработанная в



1946 г. в университете штата Пенсильвания (США), содержала более 17 тыс. электронных ламп, 70 тыс. резисторов, 6 тыс. переключателей и выполняла 5 тыс. операций в секунду.



Первая отечественная и на континентальной Европе электронная вычислительная машина (ЭВМ) - МЭСМ (Малая электронная счётная машина), была введена в регулярную эксплуатацию 25 декабря 1951 года в Феофании, г. Киев. Машина эксплуатировалась до 1957 года, а затем была передана в КПИ, где проработала до 1959 г.

Компьютеры обслуживал целый штат инженеров, необходимо было подсоединить многочисленные провода, на что уходило много времени. В 1948 г. были изобретены транзисторы - малогабаритные приборы, которые заменили в компьютерах электронные лампы. В 1958 г. Джек Килби придумал, как на одной пластине полупроводника получить несколько транзисторов, а в 1959 г. Роберт Нойс нашёл метод, позволивший создавать на одной пластине и транзисторы, и необходимые соединения между ними – родилась интегральная микросхема.

Первый персональный компьютер "Altair" появился в 1974 г. И хотя до него существовали такие модели, как Kenbak (1971 г.) и Micral (1973 г.), микрокомпьютерная революция началась именно с "Altair". Разработчик "Altair" – крохотная компания MITS из Альбукерка (штат Нью-Мексико, США) – продавала машину в виде комплекта деталей за 397 долл., а полностью собранной – за 498 долл. У "Altair" был процессор Intel 8080 с тактовой частотой 2 МГц и оперативная память в 256 байт, а клавиатура и дисплей отсутствовали.

Полюбовавшись в своем гарвардском общежитии на "Altair" с обложки журнала "Popular Electronics", Билл Гейтс с Полом Алленом адаптировали к ней язык программирования BASIC и выбыли из колледжа. Гейтс и Аллен продолжили свое дело и,

как известно, сделались мультимиллиардерами, а глава MITS Эд Робертс в 1977 г. продал свою компанию и стал сельским врачом. Два года спустя производство "Altair" прекратилось.

Первые ПК требовали от пользователей некоторых профессиональных навыков, например умения программировать или знания двоичной математики. Ситуация изменилась в 1977 г. с появлением машин, готовых к работе "прямо из коробки". "Apple II" мог гордиться дивной цветной графикой, а у PET 2001 компании Commodore был встроенный монитор. В 1981 г.,компания International Business Machines выпустил первый IBM PC с **открытой архитектурой.**

Открытая архитектура – это доступные всем технические нормативы, описывающие конструкцию компьютера, рабочие режимы, протоколы обмена данными, что предоставляет следующие возможности:

- конструкция предусматривает возможность расширения системы;
- использование технических решений и технологий не требует лицензионных затрат;
- в процессе эксплуатации возможно изменение базового состава системы самим пользователем.

Компьютерная архитектура имеет в своей основе две составляющие: аппаратную часть (железо) и программную часть (программное обеспечение). Все это носящее название ПК (состоящее из корпуса, блока питания, системной платы, винчестера, дисководов, различных периферийных устройств) является "мертвым" без соответствующего программного обеспечения, которое "оживляет" весь этот сложный набор деталей.

С развитием ПК развивается и соответствующее программное обеспечение. Для начала стала необходимость создания системного ПО, которое бы уже непосредственно управляло работой компьютера. Такой системой стала MS-DOS (MicroSoft Disk Operation System). На смену ей пришла Microsoft Windows, которая является и по сей день фактическим лидером операционных систем на рынке IBM-совместимых компьютеров.

Что такое язык программирования и компьютерные программы.

Язык программирования есть формальная знаковая система, предназначенная для записи компьютерных программ. **Компьютерная программа** - это логическая последовательность команд предназначенных для выполнения устройством (компьютером). Все языки программирования относятся к алгоритмическим языкам. **Алгоритмический язык** - есть формальный язык, используемый для записи, реализации или изучения алгоритмов. **Алгоритм** – последовательный логический порядок действий необходимых для достижения цели.

Возникновение и развитие алгоритмического языка Basic

Язык программирования Basic был создан в 1964 году двумя профессорами из Dartmouth College (Гановер, Нью-Гэмпшир, США) - Джоном Кенеми (на снимке справа) и Томасом Куртцом для обучения студентов навыкам программирования. Язык получился настолько простым и понятным, что через некоторое время его начали применять и в других учебных заведениях. В 1975 году, с приходом первых микрокомпьютеров, Билл Гейтс и Пол Аллен, основатели фирмы Microsoft, создали новую версию Basic для первых компьютеров "Альтаир" (MITS Altairs, 1974 г.).

С появлением первых персональных компьютеров IBM PC, именно он стал стандартом в программировании, но уже в виде GW-Basic. Потом был Turbo Basic, QuickBasic, Basic PDS и пр. Но всегда при разработке новой версии языка сохранялась совместимость с прежними версиями и программа, написанная для практически первого Basic, с незначительными изменениями могла работать и в последующих версиях этого языка.

Выход в свет операционной системы Microsoft Windows с новым графическим интерфейсом пользователя (Graphical User Interface, GUI), насыщенным кнопками, пиктограммами и другими трудно программируемыми объектами, привёл к появлению в 1991 г. нового инструментального средства Microsoft Visual Basic. Эта

система программирования позволяла не вдаваться в сложнейшие внутренние структуры Windows и создавать программы из "кубиков", как в детском конструкторе. Это позволило перейти от процедурного к объектно-ориентированному программированию, которое позволяет группировать функции и данные в единой сущности – "объекте", несущем определённые свойства. Меню, окна, списки, кнопки, поля ввода текста и другие элементы интерфейса (**интерфейс** – система связей для взаимодействия нескольких объектов или систем с целью обмена информацией) Windows стали добавляться в программу с помощью простейших операций.

В 1994 году, с выходом Visual Basic for Applications в составе Microsoft Office, Basic превращается в один из основных стандартов программирования для Windows. Его грамматическая простота основана на том, что первоначально он был создан как язык для обучения: "Beginner's All-purpose Symbolic Instructional Code" (Многоцелевой символьный командный код для начинающих). Программы на VB работают медленнее своих аналогов на C, но все же они достаточно быстры для многих деловых целей и требуют гораздо меньше времени на разработку. При помощи VBA можно писать сценарии для web-браузеров и автоматизировать приложения Microsoft Office.

Получив титул самого популярного языка программирования XX века, Visual Basic и в настоящее время наиболее востребован как в среде программистов, так и среди обычных пользователей персональных компьютеров.

Преимущества и недостатки, область применения VBA

Аббревиатура VBA (Visual Basic for Applications) означает Visual Basic для приложений. VBA - это язык программирования, встроенный во множество программ и приложений от Microsoft Office, Microsoft Project, Visio и AutoCAD до многочисленных специализированных приложений, предназначенных для управления производственными процессами, учета финансовых ресурсов или информационной поддержки клиентов. VBA быстро и неуклонно движется по направлению к тому, чтобы стать стандартом в индустрии создания программ. После освоения VBA вы сможете использовать этот язык в любом из приложений, поддерживающих VBA.

VBA – самый удобный язык для работы с приложениями Office. Язык VBA встроен в приложения Office (и не только), и код на языке VBA можно хранить

внутри документов приложений Office – документах Word, книгах Excel, презентациях PowerPoint и т.п. Этот код можно запускать оттуда на выполнение, поскольку среда выполнения кода VBA (на программистском сленге — хост) встроена внутрь этих приложений.

Например, в Microsoft Word нет команды для сохранения выделенного блока текста в отдельном файле. Для решения этой задачи придется скопировать выделенный текст в буфер обмена, создать новый документ, скопировать в него содержимое буфера, сохранить и закрыть этот новый документ. С помощью VBA можно создать небольшую программу, которая выполнит все эти шаги автоматически. Для этого нужно в окне документа Microsoft Word открыть редактор VBA нажав **Alt+F11**, открыть в меню окно создания модуля (**Insert→Module**) и ввести следующий текст процедуры:

Sub BlockToFile()	➢ начало процедуры с именем "BlockToFile"
Selection.Range.Copy	➢ копирует выделенное в буфер обмена
Documents.Add	➢ создает новый документ
Selection.Range.Paste	➢ вставляет содержимое буфера в новый документ
Dialogs(wdDialogFileSaveAs).Show	➢ открывает диалоговое окно "Сохранение документа"
ActiveDocument.Close	➢ закрывает новый документ
End Sub	➢ конец процедуры

Теперь **в окне исходного документа** нужно **выделить необходимый фрагмент** текста и, нажав клавиши **Alt+F8**, выполнить макрос **BlockToFile –** первая программа на VBA сделана. В среде программистов-профессионалов считается, что самый короткий путь "с нуля" и до профессиональных программ, которые делаются под заказ – именно через связку Office- VBA (и конечно, не через C++, Java или Delphi).

Макрокоманда или **макрос** — программный алгоритм действий, записанный пользователем. Другими словами, последовательностей команд, чтобы пользоваться ими снова и снова. Все приложения Microsoft Office имеют средство записи макросов.

VBA отличает от Visual Basic невозможность создавать независимые приложения и более медленная работа при первом запуске. Последнее происходит в результате необходимости компиляции. **Компиляция** – процесс перевода программного кода (который вы можете прочитать) в машинный код (представляющий собой инструкции, которые непосредственно исполняются компьютером во время выполнения программы).

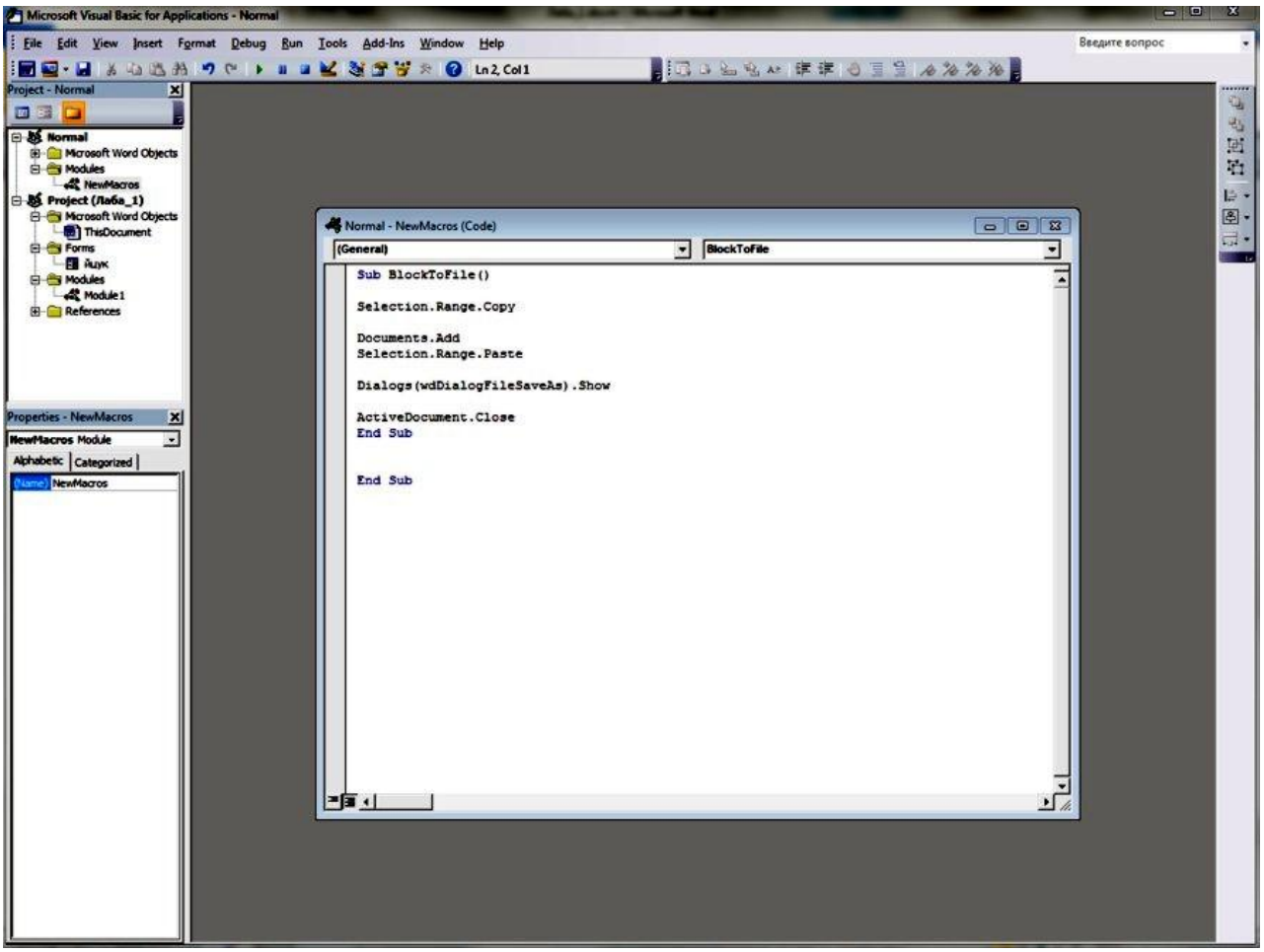


Рис.1 – Вид окна редактора VBA при записи макроса.

Контрольные понятия для изучения.

1. История развития вычислительной техники. Открытая архитектура.
2. Компьютерная программа.
3. Язык программирования.
4. Алгоритм, алгоритмический язык.

5. Интерфейс.
6. Язык Visual Basic for Applications, история создания.
7. Макрокоманда или макрос.
8. Компиляция.

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Набрать в Microsoft Word приведённый макрос и проверить его действие.
3. Отобразить в протоколе ход выполнения макроса, зарисовать открывающиеся при этом окна, перечислить основные трудности, возникшие у Вас при выполнении лабораторной работы.

Приложение

Правила оформления протокола выполнения работы.

1. Протокол выполняется на отдельных листах или в тетради, однако, однообразно у всей учебной группы.
2. Протокол должен содержать:
   - отдельную титульную страницу;
   - теоретическую часть, содержащую в конспективном виде априорную информацию, необходимую студенту для защиты лабораторной работы;
   - текст процедур (макросов) использованных в работе с подробным описанием действия операторов;
   - необходимые рисунки для наглядного объяснения порядка проведения работы;
   - результаты выполнения процедур (макросов), использованных в работе:
   - выводы по результатам работы;
   - информация о вероятных трудностях, возникших в ходе выполнения работы.
3. Титульный заголовок работы выполняется на отдельной странице.

Пример титульной страницы.

Інформатика, програмування та числові методи.

Лабораторна робота №1

Мова програмування VBA. Основні поняття та застосування.

Студента 1 курсу ІФФ НТУУ "КПІ",

група ФЛ-41

Голобородька Богдана Йосиповича

ЛАБОРАТОРНАЯ РАБОТА №2

Основы работы в среде алгоритмического языка VBA.

Цель: ознакомится с основными этапами создания программы.

Основные вопросы.

1. Редактор VBA
2. Проектирование программы
3. Реализация проекта
4. Тестирование программы
5. Отладка

Редактор VBA

Редактор Visual Basic служит командным центром для работы в VBA. В нем вы должны находиться при разработке VBA-форм, создании VBA-кода, тестировании и отладке VBA-программ. Чтобы запустить редактор в большинстве VBA-приложений версии Microsoft Office 2010 и выше можно воспользоваться одним из двух методов:

1. Выбрать из меню **Вид→Макросы→Макросы→ввести имя макроса→Создать** (*версии Microsoft Office ранее 2007 г. пункт меню* ***Сервис→Макрос→Макросы→Редактор Visual Basic***).
2. Нажать **Alt+Fll**.

Всего в редакторе VBA возможно открытие девяти типов окон. Имя типа окна расположено в скобках

Имя окна	Функциональное назначение
Project Explorer (окно проводника проекта)	Перемещение по компонентам VBA-проекта и управление ими
Code (окно программного кода)	Просмотр, добавление и редактирование программного кодаУВА
UserForm (окно формы)	Проектирование пользовательских форм (диалоговых и других окон)
Toolbox (панель элементов управления)	Добавление элементов управления (текстовых полей или кнопок} в формы, а во многих VBA-приложениях и в документы

Properties (окно свойств)	Установка индивидуальных атрибутов выделенной формы или элемента управления
Watch (окно контролируемых выражений)	Отслеживание значений выбранных переменных программы и выражений
Locals (окно локальных переменных)	Отслеживание значений переменных текущей процедуры
Immediate (окно немедленного выполнения)	Выполнение отдельных строк программного кода для немедленного получения результата
Object Browser (окно обозревателя объектов)	Исследование объектов, доступных программам

Процесс создания программы делится на несколько этапов:

1. Проектирование программы.
2. Реализация проекта.
3. Тестирование программы.
4. Отладка.

Программа-пример должна будет открывать на экране новое окно с показанной в нем надписью, а также датой и временем. Окно будет оставаться на экране до тех пор, пока пользователь не щелкнет на кнопке ОК. Такой тип надписи принято называть "**сообщение**" (**message**).

Проектирование программы.

1. Программа имеет одно окно, поэтому вам понадобится одна форма (**UserForm**).
2. Для формы потребуются два элемента управления – надпись для сообщения и кнопка для команды "**ОК**".
3. Нужно будет также создать программный код для двух процедур: одной – для надписи, в которую нужно поместить сообщение, а другой – для выхода из программы, когда пользователь щелкнет на кнопке ОК.

Это программа не требует создания отдельного модуля (**модуль** – это отдельная единица программного кода, содержащая одну или несколько процедур, **процедура** (подпрограмма) – поименованная или иным образом идентифицированная часть компьютерной программы, содержащая описание определённого набора действий). Обе наши процедуры можно разместить в окне формы, т.к. они реагируют

на события связанные с формой и, в отличии от предыдущей программы, не требуют создания отдельного модуля кода (Code).

Реализация проекта.

Добавление новой формы.

В меню редактора выберите "**Inserts→UserForm**". На экране появится серая панель **UserForm1**. Маркёрами изменения размеров можно менять габариты панели, а в окне **Properties** (по-умолчанию находится в левой части экрана) её цвет, имя и другие свойства. Одновременно с панелью формы на экране появится панель элементов управления с различными пиктограммами. Панель элементов управления видна, только когда панель формы активна.

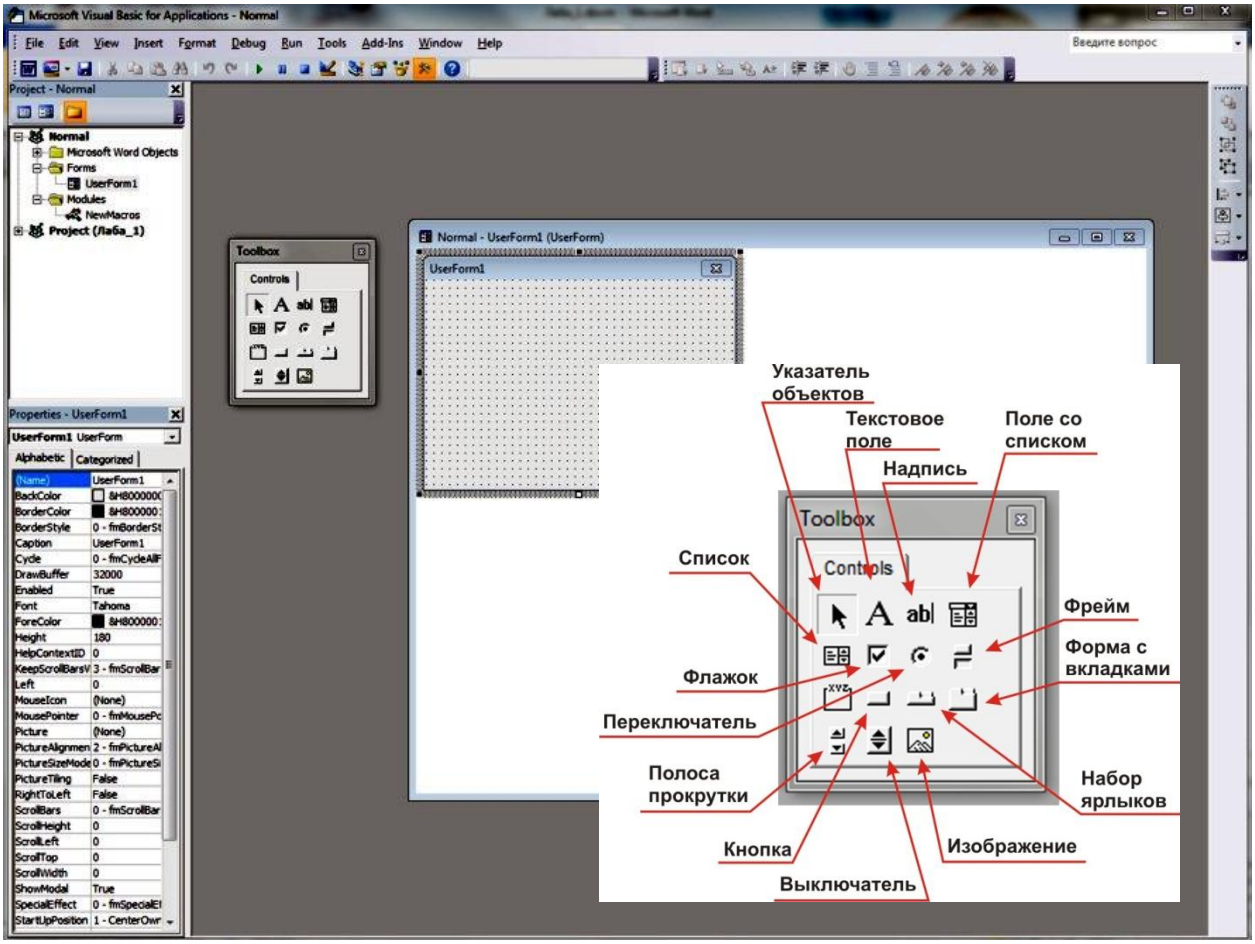


Рис. 2.1 – Вид окна редактора VBA при создании формы и расположение пиктограмм на панели управления..

Добавление надписи на форму.

Наведите указатель на пиктограмму с буквой "**A**" панели управления и переместите её на область формы. С помощью маркёров изменения размеров

придайте окну необходимую форму. В панели **Properties** удалите надпись **Label1** в свойстве **Caption**, т.к. программа сама введёт текст надписи во время выполнения.

Пиктограмма – знак, отображающий важнейшие узнаваемые черты объекта

Добавление кнопки команд.

Щёлкните на окне формы, чтобы сделать его активным. Перетащите пиктограмму кнопки на поле формы и задайте её размеры. Измените свойство **Caption** кнопки на текст "**Ok**". Установите необходимые свойства текста в поле **Font** панели **Properties**.

Таким образом, создана полностью функциональная форма для всей программы. Теперь можно перейти к созданию программного кода.

Добавление программного кода.

Программный код должен, в данной программе, содержать две процедуры. Первая процедура должна при появлении на экране формы отобразить нужное сообщение, а вторая – завершить выполнение программы, когда кто-нибудь щёлкнет на кнопке "**Ok**".

Вызов окна программного кода формы.

Для вызова окна программного кода воспользуйтесь любым из следующих способов:

1. Выбрать **View→Code** из меню.
2. Нажать "**F7**".
3. Щелкнуть на форме правой кнопкой мыши и в появившемся контекстном меню выбрать **View Code**.

В появившемся окне уже есть заготовка начала и конца частной процедуры (подпрограммы) **Private Sub** для события нажатия **Click()** кнопки **CommandButton1**. Любая подпрограмма должна заканчиваться служебным словом **End Sub**.

Напечатав строку **Unload Me** между имеющимися строками программного кода, мы дадим возможность программе завершить работу. Команда **Unload Me** убирает указанный объект из памяти.

Создание второй процедуры.

В окне программного кода, которое должно у вас остаться активным, выполните следующие шаги:

В текстовом поле списка слева вверху окна кода выберите **UserForm**. Появится заготовка процедуры, выполняемой при щелчке на любом месте формы, где нет элементов управления.

В текстовом поле списка справа вверху окна кода выберите процедуру **Activate()**, которая вызывается при событии загрузки формы в память. Теперь процедуру **UserForm_Click()** можно удалить, выделив и нажав клавишу "**Del**".

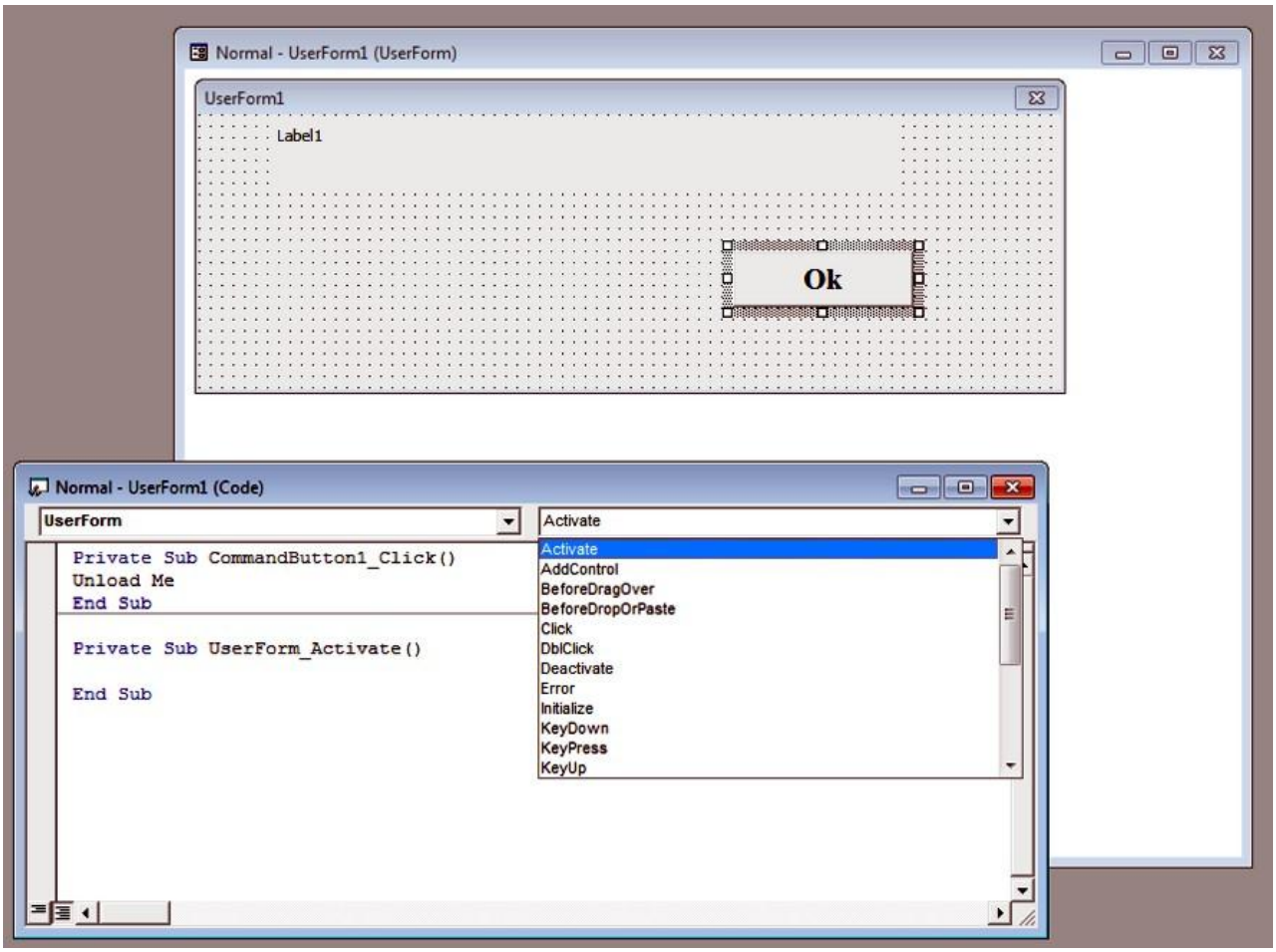


Рис 2.2 – Создание новой процедуры.

Добавление программного кода в новую процедуру.

Наполнение процедуры должно выглядеть следующим образом:

Private Sub UserForm_Activate()	Начало процедуры выполняемой при событии загрузки формы в память

Dim Chas As String	Создаётся переменная с именем Chas, как строковая, т.е. как набор символов.
Chas = " Час минає! - Звикли говорити ви внаслідок невірного поняття. Час вічний: минаєте ви!: "	Переменной Chas присваивается значение " Час минає! - Звикли говорити ви внаслідок невірного поняття. Час вічний: минаєте ви! "
Label1.Caption = Chas & Format(Now, "dddddd, hh ч. mm мин.")	Меняет свойство Caption текстового поля Label1 на значение переменной Chas и возвращает текущую компьютерную дату и время в указанном в скобках формате.
End Sub	Конец процедуры.

Тестирование программы.

Запуск программы в редакторе VBA.

Запустить программу на выполнение из редактора VBA можно тремя способами.

1. Выбрать в меню **Run** пункт **RunSub/UserForm**.
2. Щёлкнуть на кнопке **Run** панели инструментов.
3. Нажать клавишу "**F5**"

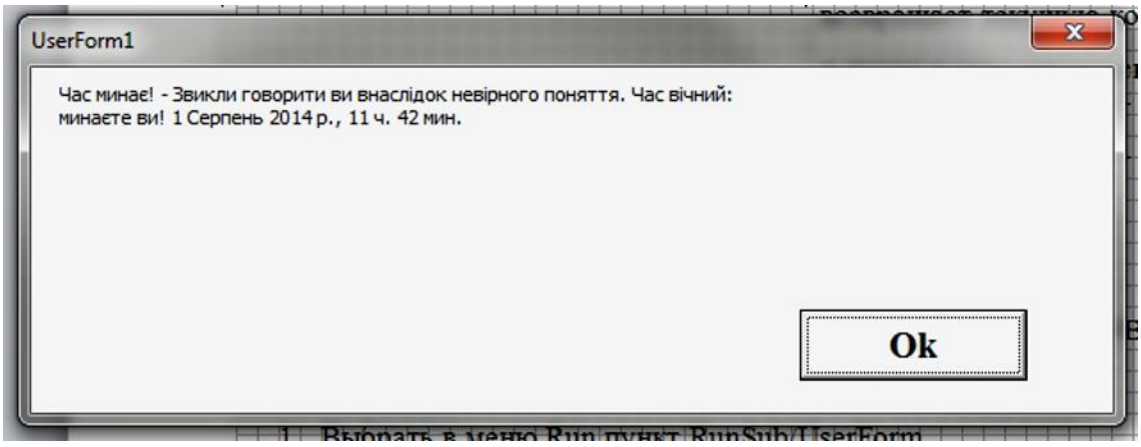


Рис.2.3 – Результат тестирования программы.

На медленных компьютерах перед первым запуском программы на VBA может произойти небольшая задержка вызванная процессом компиляции.

Отладка.

С усложнением программы увеличивается вероятность появления ошибок. В небольших программа найти ошибку помогает внимательное, шаг за шагом, изучение текста (листинга). Часть ошибок помогает выявить отладчик при компиляции программы. Однако, компилятор не может выявить большинство логических ошибок алгоритма. Поэтому, всегда нужно помнить Третий закон Грида: машинная программа выполняет то, что вы ей приказали делать, а не то, что бы вы хотели, чтобы она делала.

Контрольные понятия для изучения.

1. Модуль
2. Процедура
3. Пиктограмма
4. Форма программы
5. Программный код

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Выполнить в Microsoft Word все этапы и создать программу сообщения на VBA.
3. Изменить текст сообщения и запустить эту программу на выполнение в среде Microsoft Excel.
4. Отобразить в протоколе ход выполнения работы, текст процедур и рисунки для пояснения их работы. Описать основные трудности, возникшие у Вас при выполнении работы..

ЛАБОРАТОРНАЯ РАБОТА №3

Синтаксис языка программирования VBA.

Цель: изучить типы данных, строение и использование основных операторов языка программирования VBA.

Основные вопросы.

1. Диалоговые окна VBA.
2. Переменные. Типы данных.
3. Константы.
4. Операторы присваивания.
5. Арифметические выражения.
6. Математические функции.

Диалоговые окна VBA.

В VBA существуют две возможности создания диалоговых окон, позволяющих вести интерактивный диалог с пользователями.

Окно сообщений **MsgBox** выводит простейшие сообщения для пользователя, а окно ввода **InputBox** обеспечивает ввод информации.

Синтаксис:

InputBox (*сообщение* [, *заголовок*] [, *default*] [, *xpos*] [, *ypos*])

Аргументы:

сообщение – строковое (текстовое) выражение, отображаемое как сообщение в диалоговом окне. Может содержать несколько строк.;

заголовок - строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку помещается имя приложения;

Default - строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода отображается пустым;

Xpos - числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана;

Ypos - числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана.

Синтаксис:

MsgBox *сообщение* [, *кнопки*] [, *заголовок*] [, *файл_справки*, *раздел*]

Аргументы

сообщение - строковое выражение, отображаемое как сообщение в диалоговом окне;

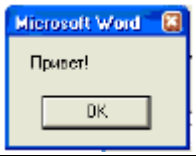
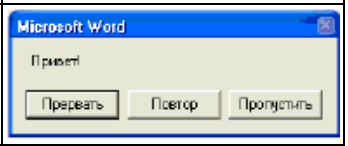
кнопки - числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию равно 0. Все значения данного аргумента указаны в таблице 3.1;

заголовок - строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку помещается имя приложения;

файл справки - строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо наличие также аргумента context;

раздел - числовое выражение, определяющее номер соответствующего раздела справочной системы.

Таблица 3.1 – Значения аргумента кнопки процедуры MsgBox

Константа	Значение / код	Отображаемые кнопки
VbOkOnly	0 MsgBox("Привет!", vbOKOnly)=vbOk	
VbOkCancel	1 MsgBox ("Привет!", vbOKCancel)=vbOk	
VbAbortRetryIgnore	2 MsgBox ("Привет!", VbAbortRetryIgnore)=vbAbort	
VbYesNoCancel	3 MsgBox("Привет!", VbYesNoCancel)=vbYes	
VbYesNo	4 MsgBox ("Привет!", vbYesNo)=vbYes	

Константа	Значение / код	Отображаемые кнопки
VbRetryCancel	5 MsgBox ("Привет!", VbRetryCancel)=vbretry	
VbCritical	16 MsgBox("Привет!", vbCritical)=vbYes	 Критическое сообщение
VbQuestion	32 MsgBox("Привет!", VbQuestion)=vbYes	 Запрос
VbExclamation	48 MsgBox("Привет!", VbExclamation)=vbYes	 Предупреждение
VbInformation	64 MsgBox("Привет!", vbInformation)=vbYes	 Информация

Если в окне сообщения всего две кнопки, для выяснения, на какой из кнопок был щелчок, прекрасно подходит оператор **If … then**. Например:

If MsgBox ("Начинать?", vbYesNo)= vbYes then

Операторы на действие этой кнопки "Yes"

Else

Операторы на действие другой кнопки

End if

Переменные. Типы данных

Переменные в программировании имеют такой же смысл, как в математике. Перед тем, как использовать переменную, ее рекомендуется описать (объявить).

Синтаксис оператора описания переменной:

Dim *переменная* [As *тип*]

в этой конструкции:

- **Dim** - ключевое слово, свидетельствующее о том, что объявляется переменная (dimension - размер);
- ***переменная*** - имя объявляемой переменной;
- **As** - ключевое слово, используемое при задании типа данных (as - как);
- ***тип*** - тип данных для объявляемой переменной или (что то же самое) тип переменной.

Здесь и далее квадратными скобками выделяется необязательная часть конструкции (которая может отсутствовать).

Другими словами, имеются две конструкции оператора описания переменной:

Dim ***переменная***

Dim ***переменная*** **As** ***тип***

Таблица 3.2 – Тип данных VBA.

Тип данных	**Размер ячейки (В - байт)**	**Значения переменной/константы или примечание**
Boolean (логический)	2B	True (истина, логическая единица) и False (ложь, логический нуль)
Byte (короткий целый беззнаковый)	1B	Целые числа от 0 до 255
Integer (целый)	2B	Целые числа от -32 768 до 32 767
Long (длинный целый)	4B	Целые числа от -2 147 483 648 до 2 147 483 647
Currency (денежный)	8B	Числа с четырьмя десятичными знаками от -922 337 203 685 477.5808 до 922 337 203 685 477.5807
Single (с плавающей точкой одинарной точности)	4B	Числа с дробной частью от $-3.402823 \cdot 10^{38}$ до $-1.401298 \cdot 10^{-45}$ для отрицательных чисел и от $1.401298 \cdot 10^{-45}$ до $3.402823 \cdot 10^{38}$ для положительных чисел

Double (с плавающей точкой двойной точности)	8B	Числа с дробной частью от -1.79769313486231• 10^{308} до -4.94065645841247•10^{-324} для отрицательных чисел и от 4.94065645841247 . 10^{-324} до 1.79769313486232• 10^{308} для положительных чисел
Date (дата)	8B	От 1 января 100 года до 31 декабря 9999 года
String (строковый переменной длины)	10B + 1B на символ	Длина строки от 0 до 2^{31} символов
String (строковый постоянной длины)	Задаётся при выполнении оператора Dim	Длина строки от 1 до 2^{16} символов
Variant (универсальный)	Для чисел 16B	Значения соответствуют типу данных Boolean, Byte, Integer, Long, Currency, Single, Double или Date, определяемому автоматически
	Для строк 22B + 1B на символ	Длина строки от 0 до 2^{31} символов
Object (объект)	4B	Используется при объявлении объектов; аналогичен типу Variant

Когда при выполнении программы компьютер встречает оператор Dim, он выделяет переменной ***переменная*** часть своей оперативной памяти, которую в программировании принято называть ячейкой. Размер выделенной ячейки, исчисляемый в байтах, определяется типом переменной ***тип*** (Табл.3.2).

Одним оператором Dim можно описать несколько переменных, перечислив их через запятую. Пример:

Dim i As Byte, j As Integer, k As Integer

Типы данных Byte, Integer, Long, Currency, Single и Double называются числовыми типами данных. Согласно третьему столбцу таблицы 3.1:

- в ячейке оперативной памяти компьютера, соответствующей переменной типа Byte, могут храниться только натуральные числа и нуль;
- в ячейке, соответствующей переменной типа Integer или Long, могут храниться целые числа;
- в ячейке, соответствующей переменной типа Currency, Single или Double, могут храниться числа с дробной частью.

Если при описании переменной программист не указывает тип данных (например, Dim W), То переменной (w) автоматически будет присвоен тип Variant. Это означает, что в ячейке, соответствующей этой переменной, Может храниться информация любого вида, т. е. тип данных Variant аналогичен формату "Общий" табличного процессора Excel.

Рассмотрим оператор

Dim i, j As Integer

Этот оператор эквивалентен следующему:

Dim i As Variant, j As Integer

Если мы хотим, чтобы обе переменные (i, j) имели тип Integer, то должны их описать так:

Dim i As Integer, j As Integer

или

Dim i As Integer

Dim j As Integer

В языках программирования **ключевыми (служебными) словами** называются такие слова, которые используются только в конструкциях языка. Ключевые слова нельзя использовать в качестве имен программ и переменных в программах. По умолчанию среда VBA настроена таким образом, что при наборе текста программы в окне кода, все ключевые слова выделяются синим цветом (комментарии выделяются зеленым цветом, синтаксические ошибки - красным).

Имя программы назначается ее разработчиком. Оно должно удовлетворять следующим условиям:

- первый символ имени должен быть буквой;
- имя может содержать только буквы, цифры и символ соединения "_";
- имя не должно содержать более 255 символов.

Подчеркнем, что в имени программы нельзя использовать пробел. Если мы хотим, чтобы имя программы состояло из нескольких слов, то вместо пробела надо использовать знак соединения или каждое слово начинать с заглавной буквы.

Ограничения на имена переменных такие же, как на имена программ, причем регистр букв не имеет значения. Например, если объявлена переменная **Alpha**, а затем

мы пытаемся объявить переменную **alpha**, то слово "**Alpha**" будет исправлено на "**alpha**". Однако, переменная "**oca"**, набранные русскими и английскими буквами, являются разными переменными, и при выполнении программы им отводятся разные ячейки оперативной памяти.

Таблица 3.3 – функции преобразования типа данных.

Функция	Возвращает тип	Действие
CBool	Boolean	Преобразует значение в булевый тип
CByte	Byte	Преобразует значение в тип Byte
CCur	Currency	Преобразует значение в тип Currency
CDate	Date	Преобразует значение в Дату и Время
CDbl	Double	Преобразует значение в тип Double
CDec	Decimal	Преобразует значение в подтип Decimal типа Variant
CInt	Integer	Преобразует значение в целый тип
CLng	Long	Преобразует значение в длинное целое
CSng	Single	Преобразует значение в тип Single
CStr	String	Преобразует значение в строку
CVar	Variant	Преобразует значение в тип Variant

Таблица 3.4 – Греческий алфавит

Строчная буква	Заглавная буква	Произношение	Латинское написание
α	A	альфа	alpha
β	B	бета	beta
γ	Γ	гамма	gamma
δ	Δ	дельта	delta
ε	E	эпсилон	epsilon
ζ	Z	дзета	zeta
η	H	эта	eta
θ	Θ	тета	theta
ι	I	йота	iota
κ	K	каппа	kappa
λ	Λ	ламбда	lambda
μ	M	мю	mu
ν	N	ню	nu

ξ	Ξ	кси	xi
ο	Ο	омикрон	omicron
π	Π	пи	pi
ρ	Ρ	ро	rho
σ	Σ	сигма	sigma
τ	Τ	тау	tau
υ	Υ	ипсилон	upsilon
φ	Φ	фи	phi
χ	Χ	хи	chi
ψ	Ψ	пси	psi
ω	Ω	омега	omega

Константы

Наряду с переменными в VBA используются константы. Как и переменной, константе соответствует ячейка оперативной памяти. Однако, в отличие от переменной, содержимое ячейки, соответствующей константе, в программе изменить нельзя.

Существует две разновидности констант - пользовательские и встроенные. Пользовательские константы требуют объявления. Для этого используется оператор вида

Const *константа* [As *тип*] = *значение*

где:

- **Const** - ключевое слово, которое показывает, что объявляется константа;
- **As** - ключевое слово, с которого начинается задание типа данных;
- ***константа*** - имя объявляемой константы;
- ***тип*** - тип данных для объявляемой константы или (что то же самое) тип константы;
- ***значение*** - значение, присваиваемое константе.

Ограничения на имена констант такие же, как на имена переменных.

Примеры объявления констант:

```
Const pi As Double = 3.141592654
Const e As Double = 2.718281828
Const Message = "Завершение работы"
Const Millennium As Date = #1 Jan 2000#
Const beta As Currency = 1/3
```

При помощи одного оператора Const можно объявить несколько констант, перечислив их через запятую:

Const Min = O, Max = 1000, Flag As Boolean = False

В качестве примера использования констант рассмотрим следующую программу перевода угла (al1g1e) из градусной меры в радианную:

```
Sub deg2rad ()
    Dim angleD As Currency
    Dim angleR As Currency
    Const pi As Double = 3.141592654
    angleD 270                      'Угол равен 270 градусам
    angleR = angleD * pi / 180      'Результат: угол в радианах
End Sub
```

Встроенные константы не требуют объявления. Для имен (в частности имен констант) разработчиками Windows принято следующее соглашение: имена данных, близких по смыслу, начинаются с одного и того же короткого префикса. Например, встроенные константы Visl1al Basic имеют префикс **vb**, встроенные константы Excel имеют префикс **xl**, встроенные константы Word – **wd** и т.д.

Операторы присваивания.

Оператор присваивания имеет следующий синтаксис:

переменная = выражение

В нем ***переменная*** - имя переменной, ***выражение*** – арифметическое или логическое выражение или строка, которую можно рассматривать как выражение.

Знак "=" в операторе присваивания называется знаком присваивания.

Оператор присваивания работает следующим образом:

- компьютер рассчитывает значение выражения выражение;
- полученное значение записывается в ячейку оперативной памяти компьютера, соответствующую переменной переменная, т. е. присваивается этой переменной.

При выполнении оператора присваивания может происходить преобразование типа данных.

Арифметические выражения

Целое число в VBA представляется последовательностью цифр со знаком "минус" или без знака. Примеры целых чисел:

-18 32 0

Если в числе имеется дробная часть, то она отделяется от целой части точкой. При этом целую часть можно опустить, если она равна нулю. Примеры чисел с дробной частью:

0.5 -5.68 -.12 3 . .03

Возможно также представление чисел с дробной частью в экспоненциальной форме. Например, заряд электрона, равный $\mathbf{1.6 \cdot 10^{-19}}$ кулона, запишется в виде **1.6E-19**

Вместо английской буквы **E** при указании порядка можно использовать букву **D**, т. е. заряд электрона можно записать в виде **1.6D-19**.

Таблица 3.5 – Арифметические операции.

+	➢ сложение,
–	➢ вычитание, изменение знака,
*****	➢ умножение,
/	➢ деление,
^	➢ возведение в степень,
****	➢ целочисленное деление, т. е. деление целых чисел с отбрасыванием остатка,
Mod	➢ определение остатка от деления целых чисел, являющегося целым числом.

При наличии в выражении нескольких арифметических операций порядок их выполнения определяется правилом приоритетов арифметических операций:

- в первую очередь выполняется возведение в степень;
- далее выполняются умножение и деление в порядке следования;
- целочисленное деление;
- операция определения остатка от деления целых чисел;
- операция изменения знака числа
- последнюю очередь выполняются сложение и вычитание в порядке следования

Для изменения последовательности операций используются круглые скобки: сначала рассчитываются значения арифметических выражений, заключенных в круглые скобки. Квадратные и фигурные скобки в конструкциях VBA не используются.

Пример:

```
Sub Арифметика2()
        Dim m As Integer, n As Integer
        Dim x As Single, y As Single
```

```
x = 3 : m = 2 : n = -1
y = (- 3) ^ m                    'Результат: y= 9
y = -(3 ^ m)                     'Результат: y= -9
y = -3 ^ m                       'Результат: y= -9
y = 10 + (x + 7 ) ^ (т + п)      'Результат: y= 20
y = 10 + x + 7 ^ m + n           'Результат: y= 61
End Sub
```

В арифметических выражениях могут присутствовать величины (переменные и постоянные) разных типов. Если тип, полученный при вычислении значения арифметического выражения в правой части оператора присваивания (справа от знака присваивания =), не совпадает с типом переменной в левой части оператора присваивания (слева от =), то при выполнении оператора присваивания производится преобразование типа.

Типичной является ситуация, когда значение арифметического выражения справа от знака присваивания имеет дробную часть, а переменная слева имеет тип **Integer** или **Long**. При этом, В ходе выполнения присваивания происходит преобразование значения согласно следующему правилу округления:

- если дробная часть значения равна 0.5, то это значение округляется до четного числа из двух ближайших целых чисел;
- в противном случае значение округляется до ближайшего целого числа

Математические функции

Таблица 3.6 – Основные математические функции VBA

Обращение к ФУНКЦИИ	Возвращаемое значение
Abs(x)	|x|
Atn(x)	arctg x
Cos (x)	cos x
Exp(x)	e^x
Fix(x)	Результат отбрасывания дробной части x
Int(x)	Наибольшее целое число, не превосходящее x
Log(x)	Натуральный логарифм ln x при x > 0
Sgn(x)	1, 0 или -1 в зависимости от знака x
Sin(x)	sin x

Sqr(x)	$\sqrt{x}$ при $x \geq 0$
Tan(x)	tg x при $x \neq \pm\pi/2$

Таблица 3.7 – Операторы VBA, позволяющие рассчитывать значения тригонометрической функции ctg x, обратных тригонометрических функций arcsin x, arccos x и arcctg x и десятичного логарифма lg x

ctg_x = Cos(x) / Sin(x)	'если Sin(x) <> 0
Const pi As Double = 3.141592654	
arcsin_x = Atn(x / Sqr(l– x ^ 2))	'если Abs (x) <> 1
arcsin_x = Sgn(x) * pi / 2	'если Abs (x) = 1
arcos_x = Atn(Sqr(l – x ^ 2) / x)	'если x <> 0
arcos_x = pi / 2	'если x = 0
arcctg_x = Atn(l / x)	'если x <> 0
arcctg_x = pi / 2	'если x = 0
lg_x = Log(x) / 2.302585093	'если x > 0

Вдобавок к функциям, приведенным в Таблице 3.7, рассмотрим функцию **Round (x [, n])**, предназначенную для округления чисел с дробной частью. Другими словами, имеются две функции округления - **Round (x, n)** и **Round (x)**.

Функция **Round (x, n)** возвращает в программу значение арифметического выражения x, округленное до n знаков после десятичной точки. Функция **Round (x)** возвращает целое число согласно правилу округления и **Round (x) = Cint(x)**.

Случайные числа рассчитываются с помощью функции **Rnd**. Перед обращением к функции **Rnd** должен находиться оператор **Randomize**, меняющий неявный аргумент функции **Rnd**.

Пример. Программа, рассчитывающая 10 случайных чисел от 0 до 1, имеет вид

```
Sub Случайные_Числа()
    ' Возвращает 10 случайных чисел
    Dim N As Long
    Dim I As Long
      N = 10
    Dim S(1 To 10) As Single
      Randomize
      For I = 1 To N
```

```
            S(I) = Rnd
            varD1 = CVar(S(I))
            MsgBox varD1, , "Случайное число"
        Next I
End Sub
```

Контрольные понятия для изучения.

1. Применение диалоговых окон VBA
2. Что такое типы данных VBA. Варианты объявления типов данных.
3. Переменные и константы, их различия.
4. Порядок действий в арифметических выражениях.
5. Основные математические функции VBA

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Создать в Microsoft Word процедуру вычисления среднего арифметического двух чисел с объявлением типов данных переменных и применением диалоговых окон **InputBox** и **MsgBox**.
3. Отладить и запустить эту программу на выполнение в среде Microsoft Word. Занести текст созданной процедуры в протокол.
4. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №4

Синтаксис языка программирования VBA (продолжение).

Цель: изучить использование основных операторов и конструкций языка программирования VBA.

Основные вопросы.

1. Логические выражения.
2. Оператор перехода.
3. Конструкции принятия решений.
4. Циклы.
5. Массивы.
6. Пользовательские процедуры.

Логические выражения.

Помимо арифметических выражений, в VBA можно использовать логические выражения (утверждения), принимающие одно из двух значений типа Boolean - True (истина, логическая единица) или False (ложь, логический нуль).

Пример:

```
Sub Логикal ()
        Dim x As Integer
        Dim y As Integer
        Dim blnA As Boolean
        x = 5
        y = 2
        blnA = x > y                    'Результат: blnA True
        blnA = x = y                    'Результат: blnA = False
End Sub
```

В программе **Логикal** два логических выражения: **x > y** и **x = y**.

Более сложные логические выражения составляются с помощью логических операций **Not, And** и **Or**.

Операция **Not** определяется следующим образом:

- если **A** равно **True**, то **Not A** равно **False**;
- если **A** равно **False**, то **Not A** равно **True**.

Она называется логическим НЕ или логическим отрицанием.

Таблица 3.8 – Определение операции **And**

A	B	A And B
True	True	True
True	False	False
False	True	False
False	False	False

Таблица 3.9 – Определение операции **Or**

A	B	A Or B
True	True	True
True	False	True
False	True	True
False	False	False

При наличии в выражении нескольких логических операций порядок их выполнения определяется следующим правилом приоритетов:

1. в первую очередь выполняется операция **Not**;
2. далее выполняется операция **And**;
3. в последнюю очередь выполняется операция **Or**.

Пример:

```
Sub Логика2 ( )
        Dim x As Double
        Dim y As Double
        Dim z As Double
        Dim blnA As Boolean
        x = 1
        y = 2.87
        z = 3.12
        blnA = (x > y) And (y < z)          'Результат: blnA = False
        blnA = x < y And y < z              'Результат: blnA = True
        blnA = x > y Or y > z               'Результат: blnA = False
        blnA = Not (x < y Or Not y < z)     'Результат: blnA = False
        blnA = Not x > y And x > y          'Результат: blnA = False
        blnA = Not (x > y And x > y)        'Результат: blnA = True
End Sub
```

Оператор перехода

Для изменения последовательности выполнения операторов (т. е. для ветвления программы) используется оператор перехода **GoTo**, имеющий следующий синтаксис:

GoTo ***метка***

В этом операторе ***метка*** - это целое неотрицательное число без знака (0, 1, 2, 3, ...) или последовательность букв и цифр, начинающаяся с буквы (например, start53a).

Перед оператором, на который должен быть осуществлен переход (или, что то же самое, должно быть передано управление), ставится метка с двоеточием. После выполнения оператора с меткой выполняется оператор, следующий за ним.

Если меткой является целое неотрицательное число, то это число еще называют номером оператора (строки).

Конструкции принятия решений

Для принятия решений в VBA используются операторы условного перехода. Операторы условного перехода - это инструкции, которые определяют ход выполнения других операторов программы в зависимости от результатов анализа некоторых условий. К операторам условного перехода относятся операторы ветвления **IF...Then** и выбора **Select Case**. Существует краткая (**IF...Then…Else)** и полная форма (**IF...Then...ElseIf…Else...End If**) операторов ветвления.

Общий вид краткой формы оператора ветвления и одну строку:

IF ***условие*** **Then** ***оператор*** **[Else** ***оператор*****]**

При записи инструкции в одну строку ключевые слова **End If** не применяется.

Форма условного перехода IF...Then...ElseIf…Else...End If

IF ***условие1*** **Then**
операторы
…….
[ElseIf ***условие2*** **Then]**
операторы
…….
[ElseIf ***условие3*** **Then]**
операторы
……
[Else]
операторы
…….

End If.

Если ***условие1*** истинно, то выполняется операторы следующие за ним, а затем первый, из операторов следующий за ключевым словом **End if**. Если ***условие1*** ложно, то выполняются операторы **ElseIf** в порядке их следования, чтобы проверить их условия. Если обнаружится, что одно из условий имеет значение **True** (справедливо), то выполняется оператор, который следуют непосредственно за соответствующим ключевым словом **Then**.

Если все условия ложны, то выполняется оператор, следующий за ключевым словом **Else**. Таким образом, если первое условие ложно, операторы **ElseIf** обеспечивают проверку дополнительных условий с целью выбора одного из нескольких значений.

Синтаксис оператора условного перехода Select Case имеет вид:

Select Case ***значение***
Case ***условие1***
операторы1
……
Case ***условие2***
операторы 2
…….
Case ***условиеL***
операторы L
…….
Case Else
операторы
…….
End Select.

В этом условном операторе ***значение*** переменной сравнивается с каждым из значений, которые хранятся в выражениях каждого из условий. Если значение переменной удовлетворяет одному из значений условия, будут выполняться операторы, следующие за этим условием. Затем будет выполняться оператор, следующий за ключевым словом **End Select**. Если значение переменной не удовлетворяет ни одному из значений условий, то выполняется оператор, следующий за оператором **Case Else**.

Пример применения оператора **IF...Then...ElseIf…Else...End If**:

```
Sub massege()
Dim A As String
```

```
Повтор:
A = InputBox("Введіть ПАРОЛЬ", "Введення пароля")
If A = "КПІ" Then
      MsgBox "Приємного навчання!", , "Вхід вільний"
   Else
      MsgBox "Помилка", vbCritical
      GoTo Повтор
End If
End Sub
```

Пример применения оператора **Select Case**:

```
Public Sub Вибор_Case()
Dim A As String, B As String, C As String, D As String
Повтор:
A = InputBox("Вгадайте моє ім'я?", "Введіть ім'я")
Select Case A
   Case "ІФФ"
      MsgBox "Вгадали!", vbExclamation
   Case "Іван"
      MsgBox "Не вгадали, я не Іван", vbCritical
   Case "Петро"
      MsgBox "Не вгадали, я не Петро", vbCritical
   Case Else
      MsgBox "Не вгадали, спробуйте ще раз!", vbCritical
      GoTo Повтор
End Select
End Sub
```

Циклы

Какие либо действия процедуры повторяющиеся заданное количество раз или пока выполняется или не выполняется некоторое условие называют **циклом**. Процесс выполнения все операторов, заключенных в структуру цикла, один раз называется **итерацией** цикла.

Структуры цикла, всегда выполняющиеся заданное количество раз, называются **циклами с фиксированным числом итераций**. Другие типы структур цикла повторяются переменное количество раз в зависимости от некоторого набора условий. Такие циклы называются **неопределенными циклами.**

Блок операторов, находящийся между началом и концом цикла называется "**тело цикла**".

Циклы могут вкладываться друг в друга, но не могут пересекаться.

Оператор цикла For… Next.

Синтаксис:

For ***переменная_цикла*** **= старт To** ***конец*** **[Step** ***размер*****]**
операторы
…….
Next [*переменная***]**

где:

переменная_цикла - любая численная переменная VBA

старт – любое численное выражение, определяет начальное значение для ***переменной_цикла***;

конец – численное выражение, определяет конечное значение для ***переменной_цикла***;

операторы - один, несколько или ни одного оператора VBA (тело цикла);

размер – шаг изменения ***переменной_цикла.***

По умолчанию VBA увеличивает ***переменную_цикла*** на **1** каждый раз при выполнении операторов в цикле. Можно задать другое значение (***размер*** - любое численное выражение), на которое будет изменяться ***переменная_цикла***.

Конструкции Do While и Do Until.

Когда мы не знаем точно, сколько раз должна быть выполнена та или другая команда, используются конструкции **Do While…Loop** и **Do Until…Loop**.

Конструкция **Do While** означает: выполнять какое-либо действие до тех пор, пока условие истинно. Второй вариант – **Do Until**. Все выглядит точно так же, за одним исключением: цикл будет выполняться до тех пор, пока условие ложно.

Вычисление суммы десяти цифр с помощью **Do While…Loop**:

```
Do While MyVar < 10
        MyVar = MyVar + 1
        MsgBox “ MyVar = “ & MyVar
Loop
```

Тоже с помощью **Do Until…Loop**:

```
Do Until MyVar >= 10
        MyVar = MyVar + 1
        MsgBox “ MyVar = “ & MyVar
Loop
```

Можно переписать цикл так, чтобы условие проверялось после завершения цикла:

```
Do
        MyVar = MyVar + 1
        WScript.Echo “MyVar = “ & MyVar
Loop While MyVar < 10
```

Конструкция While … Wend.

В VBA имеется также конструкция While … Wend. Это – вариант цикла, который оставлен для обратной совместимости с первыми версиями Visual Basic. Функциональные возможности – те же, что и у конструкции Do…While:

```
While My Var < 10
        MyVar = MyVar + 1
        WScript.Echo “MyVar = “ & MyVar
Wend
```

Контрольные понятия для изучения.

1. Что такое логические выражения?
2. Операторы ветвление программы.
3. Конструкции принятия решений..
4. Циклические операции в VBA.

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Для студентов с номером по списку кратному трём, создать в Microsoft Word процедуру вычисления среднего арифметического для произвольно задаваемого числа чисел с объявлением типов данных переменных, применением диалоговых окон **InputBox** и **MsgBox** и использованием оператора **For … Next**.
3. Для студентов с номером по списку кратному четырём (кроме номеров кратных трём), создать в Microsoft Word процедуру вычисления произвольного (и неполного – *факториал числа, не начинающийся с единицы*) факториала для произвольно задаваемого числа значений с объявлением типов данных переменных, применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do While … Loop**.

4. Оставшимся студентам с нечётными номерами создать в Microsoft Word процедуру вычисления произвольного (и неполного) факториала для произвольно задаваемого числа значений с объявлением типов данных переменных, применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **While … Whend**.
5. Остальным студентам создать в Microsoft Word процедуру вычисления произвольного (и неполного) факториала для произвольно задаваемого числа значений с объявлением типов данных переменных, применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do Until … Loop**.
6. Отладить и запустить эти программы на выполнение в среде Microsoft Word. Занести текст созданных процедур в протокол.
7. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №5

Синтаксис языка программирования VBA (продолжение).

Цель: изучить использование основных элементов языка программирования VBA.

Основные вопросы.

1. Понятие массива.
2. Многомерность массивов.
3. Работа с фиксированными и динамическими массивами.
4. Макрофункции для работы с массивами.
5. Пользовательские процедуры.

Массивы

Массив - это переменная, содержащая несколько значений и являющаяся по своей сути пронумерованной группой значений одного и того же типа. Массивы бывают нескольких типов, одномерными и многомерными.

Одномерный массив - это пронумерованный список значений, где каждый элемент этого списка имеет свой уникальный номер. **Уникальный номер** - это неповторяющийся номер (**индекс**) каждого элемента массива. В VBA одномерный массив можно представить в виде строки.

Многомерный массив - это переменная, которая содержит в себе набор списков. В языке VBA допускается описание массивов, имеющих до 60 размерностей, но, как правило, используются двухмерные массивы. В VBA двухмерный массив можно представить в виде строк и столбцов, поэтому их довольно часто называют таблицами или матрицами.

Прежде чем использовать массив его необходимо описать. Так как массив является переменной, то действия с массивом во многом похожи на действия с обычными переменными. В массивах используются те же типы данных, что и в обычных переменных, причём массив должен содержать только однотипные данные, однако его ограничение можно обойти, если создать массив типа **Variant** (*существует функция* ***Array****, которая позволяет создавать массив в ходе выполнения программы, без предварительного описания*).

При об'явлении массива необходимо учитывать, что массивы делятся также на фиксированные и динамические. **Фиксированный массив** - это массив с заданным размером, который в свою очередь определяется количеством элементов. **Динамический массив** - это массив с переменным размером, т.е. количество элементов может изменяться во время выполнения программы.

Объявление одномерного фиксированного массива:

```
Dim iString(3) As String
```

Объявление двухмерного фиксированного массива:

```
Dim iData(5, 2) As Double
```

Объявление динамического массива:

```
Dim iWhatIs() As Integer
```

При объявлении (описании) динамического массива его размер не указывается. В процессе выполнения программы его размер может изменяться, причём неоднократно. Поэтому динамический массив применяют, если предполагается, что размер массива не будет постоянным.

ReDim - инструкция, которая позволяет изменять динамический массив.

```
Dim iWhatIs() As Integer
ReDim iWhatIs(5)
ReDim iWhatIs(10)
```

Элементы массива, которые были созданы ранее, после использования ReDim не сохраняются.

ReDim Preserve - инструкция, которая позволяет изменять динамический массив, с сохранением всех элементов массива.

```
Dim iWhatIs()
ReDim Preserve iWhatIs(2)
iWhatIs(0) = 1
iWhatIs(1) = 2
iWhatIs(2) = 3
ReDim Preserve iWhatIs(3)
iWhatIs(3) = 4
```

Номер первого элемента по умолчанию начинается с **0**, для того чтобы изменить нумерацию используйте инструкцию **Option Base** или используйте явное указание номера первого элемента. Инструкцию **Option Base** необходимо расположить в самом начале модуля VBA.

Номер первого элемента всех массивов начинается с **1**:

```
Option Base 1
Dim iString(3) As String
Dim iData(5, 2) As Double
```

Объявление фиксированных массивов с явным описанием нижней границы :

```
Dim iString(1 To 3) As String
Dim iData(1 To 5, 1 To 2) As Double
```

Макрофункции для работы с массивом :

Array - функция позволяет создавать массив в ходе выполнения программы, без предварительного описания.

```
iArray = Array("ivan", "john", "maurizio")
```

Erase - функция используется для удаления данных, хранимых в элементах массива. Массив фиксированного размера очищается полностью, но память, отведённая под его хранение, остаётся за ним. Динамический же массив уничтожается полностью.

```
Erase iString
Erase iData
Erase iWhatIs
```

IsArray - функция проверяет, является ли переменная массивом. **IsArray** имеет всего один аргумент (переменную) и в зависимости от результатов проверки возвращает **True** - если переменная является массивом, или **False** - если нет.

Пример проверки динамического массива :

```
Dim iArray()
ReDim Preserve iArray(1 To 3)
iArray(1) = "ivan"
iArray(2) = "john"
iArray(3) = "maurizio"

ReDim Preserve iArray(1 To 4)
iArray(4) = "it's very good man's"

If IsArray(iArray) = True Then
                MsgBox "iArray - это массив", , ""
        Else
                MsgBox "iArray - это не массив", , ""
End If
```

Пример проверки массива, созданного с помощью функции Array :

```
iArray = Array("ivan", "john", "maurizio")
If IsArray(iArray) = True Then
        MsgBox "iArray - это массив", , ""
    Else
    MsgBox "iArray - это не массив", , ""
End If
```

LBound - функция определит нижнюю границу массива.

UBound - функция определит верхнюю границу массива.

```
Dim iData(5 To 15, 1 To 100) As Double
iLBound = LBound(iData)
iUBound = UBound(iData)
MsgBox "Нижняя граница массива : " & iLBound & Chr(10) _
& "Верхняя граница массива : " & iUBound, , ""
```

Применение функций **LBound**, **UBound** к обычной переменной или динамическому массиву, который ещё не был описан инструкцией **ReDim**, вызовет ошибку. Не забывайте проверять переменную, используя функцию **IsArray**.

Контрольные понятия для изучения.

1. Понятие массив. Многомерность массивов.
2. Фиксированные и динамические массивы.
3. Макрофункции для работы с массивами.

Порядок выполнения.

8. Изучить теоретическую часть и занести в протокол основные положения.
9. Для студентов с номером по списку кратному трём, создать в Microsoft Word процедуру сортировки фиксированного числового массива (максимально до 100 элементов) по убыванию с применением диалоговых окон **InputBox** и **MsgBox** и использованием оператора **For … Next**.
10. Для студентов с номером по списку кратному четырём (кроме номеров кратных трём), создать в Microsoft Word процедуру сортировки произвольного одномерного динамического числового массива по возрастанию с применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do While … Loop**.
11. Оставшимся студентам с нечётными номерами создать в Microsoft Word процедуру вычисления суммы положительных чисел числового массива с

применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **While … Whend**.

12. Остальным студентам создать в Microsoft Word процедуру вычисления произведения неположительных чисел произвольного числового массива с применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do Until … Loop**.
13. Отладить и запустить эти программы на выполнение в среде Microsoft Word. Занести текст созданных процедур в протокол.
14. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

КОНТРОЛЬНАЯ РАБОТА №1

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

1. Обчислити $f=\sum_{i=0}^{10}\left(a_i^{\,2}+56*c_i*fg_i\right)$

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

2. Обчислити $F=\prod_{i=1}^{5}(a_i+b_i)$

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

3. Обчислити H=10!*a23

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

4. Обчислити та вивести на друк всі значення $K=a^2+2*a*b*c^3$ в одному вікні, де ***a***-змінюється в діапазоні від –4 до 18 з кроком 1.

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

5. Обчислити $Y=\sum_{a=2}^{10}\left(a^2+a^3\right)$

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

6. Обчислити $Y=\prod_{a=1}^{10}\left(a^4+a\right)$

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

7. Обчислити Y=r+k*x/6!

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

8. Обчислити $Y=\prod_{a=1}^{40}\left(a^4+a\right)$, де ***a*** змінюється з кроком 4

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

9. Обчислити $f=\sum_{i=0}^{50}\left(a_i{}^2+56*c_i*f*g_i\right)$, де ***i*** змінюється з кроком 5

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

10.Вивести на друк всі значення $y_i = a_i + b_i$, де ***i*** змінюється від 5 до 10

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

11. Обчислити $F = \sum_{i=1}^{10}(a_i + b_i)^2$

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

12.Обчислити T=5!+12!

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

13.Обчислити E=5!/9!

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

14.Обчислити $Y=\prod_{a=1}^{10} a^4 + 5!$

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

15.Обчислити $f=\sum_{i=0}^{50} a_i^{\,2} + 6!$

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

16.Обчислити $J=\frac{1}{n_1}+\frac{1}{n_2}+...+\frac{1}{n_m}$, де **m**=25

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

17.Обчислити $Kl=\frac{n_1 * n_2 * ... n_m}{n_1 + n_2 + ... + n_m}$, де **m**=45

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

18.Обчислити $Z=\frac{\sum_{a=1}^{5} a^3}{\prod_{b=1}^{5} b^6}$

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

19.Знайти корені рівняння $2*x^2+4*x-48=0$

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

20.Знайти шість значень виразу $y=2a^2+x$ для $a>10$, та $y=2a^2-x$ у іншому

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

21. Обчислити w=(t*4-24*x)/(25*x-t*f), передбачити запобігання діленню на нуль.

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

22. Обчислити п'ять значень $y=k*x^2$, якщо 3 <x<7. В іншому разі обчислити $y=k*1/x^2$.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

23. Обчислити шість довільних значень $y=12*x^2*\lg(12*x)$, передбачити запобігання від'ємного значення під знаком логарифма.

Практична робота №1 з курсу програмування до першої атестації першого семестру

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

24. Обчислити $k=\sqrt{\frac{d=b-kj}{23*gf+6*vc}}$, передбачити всі випадки неможливість обчислення значень.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

25. Обчислити об’єм циліндру або довжину кола за запитанням. Передбачити неможливість обчислення у разі від’ємного значення діаметру.

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

26. Обчислити швидкість автомобіля для першої половини шляху і прискорення – для другої.

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

27.Порахувати всі парні члени довільного одновимірного масиву.

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

28.Знайти суму індексів всіх непарних членів одновимірного масиву.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

29.З довільного одновимірного масиву утворити два так, щоб один містив всі парні члени вихідного масиву, а другий – непарні.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

30. Знайти суму квадратів найбільшого та найменшого членів довільного одновимірного масиву.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

31. Довільний одновимірний масив перетворити так, щоб його членами була різниця між середньоарифметичнім усіх членів та текучим членом.

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

32.В одному циклі обчислити значення функції **F(n) = 1!*2!*3!...*n!., где n!=1*2*3*…*n.**

Практична робота №1 з курсу програмування до першої атестації першого семестру[***]

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

33. Для одновимірного довільного масиву знайти суму квадратів різниць його членів від середньоарифметичного всіх членів масиву.

Практична робота №1 з курсу програмування до першої атестації першого семестру***

Студента групи ФЛ-____

_____

(*прізвище, ім'я та по батькові*)

34. Вивести на екран в одному вікні всі члени довільного одновимірного масиву кратні трьом.

ЛАБОРАТОРНАЯ РАБОТА №6

Разбор работы программы сортировки двумерного массива.

Цель: изучить использование основных элементов языка программирования VBA в среде Microsoft Word.

Основные вопросы.

1. Ввод-вывод набора данных в одном окне редактора Microsoft Word.
2. Вложенные циклические операции.
3. Работа с двумерными массивами.
4. Разбор работы программы сортировки двумерного массива.

ПРОГРАМА СОРТУВАННЯ ДВОВИМІРНОГО МАСИВУ.

```
Sub Двовимірний_за_зростанням()

Dim n, m, i, j, C As Integer, k As Single
Dim Vih(), z As Double, Text, Text_1 As String
n = InputBox("Введіть кількість членів масиву першого напрямку", n)
m = InputBox("Введіть кількість членів масиву другого напрямку", m)

'Введення масиву
ReDim Vih(1 To n, 1 To m)
For i = 1 To n
   For j = 1 To m
      Vih(i, j) = InputBox("Введіть  " & j & "член  " & i & "рядка", "Введення масиву", Vih(i, j))
      Text = Text + CStr(Vih(i, j)) + ","
      Vih(i, j) = CDbl(Vih(i, j)) ' Звернути увагу на обов'язкову зміну типу даних!!!
   Next j
   Text = Text + Chr(10)
Next i
Text = "Вихідний масив" + Chr(10) + Text

' Сортування масиву
For i = 1 To n
Повтор:
   k = 0
   For j = 1 To m - 1
      If Vih(i, j) > Vih(i, j + 1) Then
         z = Vih(i, j)
         Vih(i, j) = Vih(i, j + 1)
         Vih(i, j + 1) = z
         k = k + 1
      End If
    Next j
    If k > 0 Then GoTo Повтор:
Next i

'Вивід результату
For i = 1 To n
```

```
    For j = 1 To m
        Text_1 = Text_1 + CStr(Vih(i, j)) + ","
    Next j
    Text_1 = Text_1 + Chr(10)
Next i
Text_1 = "Результуючий масив" + Chr(10) + Text_1
MsgBox Text & Chr(10) & Text_1
End Sub
```

Контрольные понятия для изучения.

1. Особенности применения функций MsgBox и InputBox.
2. Вложенные циклы.
3. Методы сортировки массивов данных.

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Разобрать работу программы изложенной в данной работе, занести её в протокол и запустить на выполнение
3. Проанализировать работу функций **InputBox** и **MsgBox** в случае вывода совокупности данных в одном окне Microsoft Word.
4. Изучить методы сортировки двумерных массивов данных.
5. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №7

Синтаксис языка программирования VBA (продолжение).

Цель: изучить использование основных элементов языка программирования VBA.

Основные вопросы.

5. Пользовательские процедуры.
6. Фактические и формальные параметры.
7. Описание пользовательской функции.
8. Описание пользовательской подпрограммы.

Пользовательские процедуры

Блок операторов, предназначенный для многократного выполнения в разных точках программы, целесообразно оформить как процедуру. При этом блок записывается один раз и ему присваивается имя с параметрами (**формальными**). Эта запись блока операторов называется **описанием процедуры.**

. В общем случае процедура имеет:

- входные параметры, которые считаются заданными;
- выходные параметры, рассчитываемые в ходе выполнения блока операторов.

После того, как произведено описание процедуры, в программу помещаются обращения к этой процедуре с нужными параметрами (**фактическими**). Эти обращения помещаются в те точки программы, в которых по смыслу должен присутствовать блок операторов, оформленный как процедура.

Процедуры делятся на функции и подпрограммы. Функции можно использовать в арифметических и логических выражениях и строках (т. к. функция принимает значение); подпрограммы в выражениях и строках использовать нельзя. В этом состоит основное отличие функций от подпрограмм.

Описание пользовательской функции имеет следующий синтаксис:

Function ***название*** **(*формальные_параметры*) [As *тип*]**
    операторы
    ……
End Function

где ***название*** - имя функции; ***формальные_параметры*** – имена параметров, перечисленные через запятую; ***тип*** - тип значения функции; ***операторы*** - блок операторов.

В блоке ***операторы*** обязательно должен присутствовать хотя бы один оператор присваивания, в левой части которого (слева от знака присваивания =) находится имя функции ***название***.

Обращение к функции (находящееся в программе) имеет вид:

название (фактические_ параметры)

где ***фактические_параметры*** - массивы, переменные, константы, числа и/или строки, перечисленные через запятую.

В результате обращения к функции название в программу возвращается значение этой функции, соответствующее заданным параметрам.

```
Sub Program1 ()
Dim L As Long
Dim W As Double
L = Fact(12)
W = 4.2 + Fact(10) / 2
End Sub

Function Fact(N) As Long
Dim I As Byte
Dim J As Long
J = 1
For I = 1 To N
        J = J * I
Next I
Fact = J
End Function
```

Первая группа операторов - это программа **Program1**. Вторая группа операторов - описание функции **Fact**, которая рассчитывает факториал целого положительного числа **N**, являющегося формальным параметром.

В программе два обращения к функции **Fact**, С фактическими параметрами **12** и **10.**

1. Обращение к функции **Fact** фигурирует в правой части оператора присваивания **L=Fact(12)**. в результате выполнения этого оператора значение функции **Fact** при **N=12** (Т. е. значение, возвращаемое функцией **Fact** в программу) присваивается переменной **L.**

2. Обращение к функции **Fact** фигурирует в арифметическом выражении **4 .2 + Fact(10)/2**. Значение этого арифметического выражения присваивается переменной **W.**

Описание пользовательской подпрограммы имеет следующий синтаксис:

Sub ***название*** **(*формальные_параметры*)**

операторы

End Sub

где ***название*** - имя подпрограммы; ***формальные_параметры*** – имена параметров, перечисленные через запятую; ***операторы*** – блок операторов.

Имеется два эквивалентных оператора обращения к подпрограмме:

Call ***название (фактические_параметры)***

название фактические_параметры

где ***фактические_параметры***- список фактических параметров, как в обращении к функции. При наличии ключевого слова **Call** список ***фактические_параметры*** заключается в скобки, в отсутствие **Call** скобки не ставятся.

Пример кода программы и описания подпрограммы, находящихся в одном модуле:

```
Sub Program2()
        Dim aa As Single
        Dim bb As Single
        Dim eel As Single
        Dim ec2 As Single
        Dim ce3 As Single
        aa = 3
        bb = 4
Call Hypotenuse(aa, bb, cc1)     'l-e обращение к подпрограмме
Call Hypotenuse(3, 4, cc2)       '2-е обращение к подпрограмме
Hypotenuse aa, bb, cc3           '3-е обращение к подцрограмме
End Sub


Sub Hypotenuse (ByVal A, ByVal B, ByRef C)
        C = Sqr(A^2+B^2)
End Sub
```

Первая группа операторов - это программа **Program2**, вторая - описание подпрограммы **Hypotenuse**, рассчитывающей длину гипотенузы прямоугольного треугольника. В программе **Program2** имеется три оператора обращения к

подпрограмме **Hypotenuse**, причем два из них содержат ключевое слово **Call**, а одно не содержит.

Формальные параметры **A** и **B** (в описании подпрограммы) являются входными. Это - длины катетов. Ключевое слово **ByVal** перед **A** и **B** в первой строке подпрограммы означает, что эти параметры вызываются по значению (**value** - значение). В этом случае при обращении к подпрограмме ей передаются значения **A** и **B**. Это - соответственно **3** и **4** при всех трех обращениях к подпрограмме **Hypotenuse.**

Формальный параметр **C** является выходным. Это - длина гипотенузы. Ключевое слово **ByRef** перед **C** в первой строке подпрограммы означает, что параметр **C** вызывается по ссылке (**reference** - ссылка). В этом случае при обращении к подпрограмме ей передается адрес ячейки оперативной памяти, соответствующей переменной **cc1** (при первом обращении к подпрограмме **Hypotenuse**, **cc2** (при втором обращении) или **cc3** (при третьем обращении).

Ключевое слово **ByRef** можно опускать.

Если параметрами подпрограммы являются массивы, то все они (и входные, и выходные) вызываются по ссылке.

```
        Dim N1 As Integer
Sub Program3 ()
        Dim xx(50) As Double, yy(50) As Double
        Dim i As Integer
        N1 = 3
        For i = N1 To 30
                xx(i) = 0.1 * i
        Next i
        Call XSINX(30, xx, yy)                  'обращение к подпрограмме
End Sub

Sub XSINX(ByVal N2, ByRef X() As Double, ByRef F() As Double)
        Dim j As Integer
        For j = Nl To N2
                F(j) = X(j) * Sin(X(j))
        Next j
End Sub
```

Объявление **N1** перед процедурой сделано для того, чтобы переменная **Nl** была "видима" (в плане возможности использования) и в программе, и в подпрограмме. Переменные можно объявлять в двух местах:

- внутри программы или процедуры;

- в верхней части окна кода, которая называется областью общих объявлений модуля.

Если переменная объявлена в процедуре (как переменная j в последнем коде), то только эта процедура ее видит. Другие процедуры (если они есть) и программа не могут использовать значение этой переменной и менять его. Такую переменную называют **локальной**. Говорят также, что переменная видима на уровне процедуры.

Чтобы значение переменной было доступно всем процедурам данного модуля, ее надо объявить в области общих объявлений модуля (как объявлена переменная **Nl** в последнем коде). Тогда программа и все процедуры, определенные в данном модуле, могут использовать значение этой переменной и менять его. Такую переменную называют **глобальной**. Говорят также, что переменная видима на уровне модуля.

Сказанное относительно переменных относится и к пользовательским константам, но значение константы, естественно, нельзя менять.

Объявление необязательности параметра осуществляется при помощи ключевого слова **Optional**, которое ставится перед именем этого параметра в первой строке описания процедуры. Необязательный параметр должен иметь тип **Variant.**

Пример:

```
Sub Program4 ()
        Dim vntA As Byte
        Dim vntB As Byte
        Dim C As Integer
        vntA = 5
        vntB = 10
        C = Apt(vntA, vntB)                 'Результат: vntB = 6, C = 625
        vntB = 10
        C = Apt(vntA)                       'Результат: vntB = 10, C = 625
End Sub

Function Apt(ByVal a, Optional b As Variant)
        If Not IsMissing(b) Then b = a + 1
        Apt = a^4
End Function
```

В этом примере параметр **b** функции **Apt** является необязательным, о чем говорит **Optional** перед **b**. Словосочетание **As Variant** после **b** можно опустить.

В описании функции **Apt**:

• **Not** - логическая операция;

• **IsMissing** -функция.

Значение **IsMissing(b)** равно **True** при отсутствии второго параметра в обращении к **Apt,** и **False** при наличии этого параметра.

Существует возможность указывать значение, которое должен принимать необязательный параметр при его отсутствии в обращении к процедуре.

Пример:

```
Sub Program5 ()
Dim vntA As Byte
Dim vnt~ As Byte
Dim vntC As Byte
vntA = 5
vntB = 10
Call Opt(vntA, vntC, vntB)          'Результат: vntC = 15
vntB = 10
Call Opt(vntA, vntC)                'Результат: vntC = 8
End Sub

Sub Opt(ByVal a, c, Optional b = 3)
        c = a + b
End Sub
```

В приведенном примере параметр **b** подпрограммы **Opt** является необязательным, причем, если в обращении к **Opt** присутствуют только два фактических параметра, то при выполнении оператора **c = a + b** полагается **b = 3**.

Контрольные понятия для изучения.

4. Пользовательские функции.
5. Пользовательские процедуры.
6. Понятие формальных и фактических параметров.

Порядок выполнения.

6. Изучить теоретическую часть и занести в протокол основные положения.
7. Переделать процедуры созданные в работе №4 с применением пользовательских процедур или функций и применением диалоговых окон **InputBox** и **MsgBox**.
8. Отладить и запустить эти программы на выполнение в среде Microsoft Word. Занести текст созданных процедур в протокол.
9. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №8

Понятие объектно-ориентированного языка программирования. Объекты и события.

Цель: изучить основные понятия объектно-ориентированного программирования и их применение.

Основные вопросы.

1. Понятие объекта.
2. Классы объектов и отдельные объекты.
3. Объектная модель.

Понятие объекта.

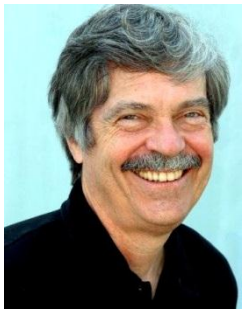
Все современные алгоритмические языки, в частности Visual Basic и VBA, являются объектно- и событийно-ориентированными. Простейший способ понять суть объектов - рассматривать их как часть приложения VBA и документов. Ячейка рабочего листа Excel является объектом, так же как и именованный диапазон ячеек. То же касается отдельных рабочих листов и рабочих книг, в которые входят ячейки, диапазоны и листы. Во всех офисных приложениях VBA панели управления и меню (так же, как кнопки и элементы меню) являются объектами.

Очевидно, что у объектов VBA существует определенная иерархия, в которой объекты одного типа содержат объекты других типов.

Объект - это именованный элемент, обладающий:

- свойствами (настройки, которые может проверять и изменять пользователь);
- методами (действия, которые может выполнять объект по запросу программы); а также в некоторых случаях
- событиями (то, что происходит с объектом и на что он может отреагировать, автоматически предпринимая заранее заданные действия).

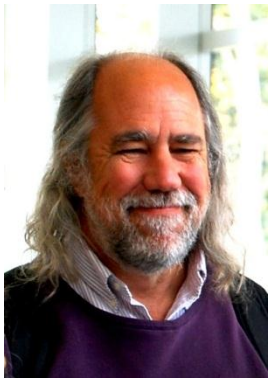
Объект - это именованный элемент программы, который содержит информацию и код, обрабатывающий эту информацию. Объект, говорят, инкапсулирует информацию и соответствующий код. **Инкапсуляция** - объединение данных с процедурами и функциями для создания нового типа данных - объекта.



Алан Кёртис Кей (*Alan Curtis Kay - американский учёный в*

области теории вычислительных систем. Один из пионеров в областях объектно-ориентированного программирования и графического интерфейса) в свое время вывел пять основных черт объекно-ориентированного языка программирования:

1. **Все является объектом.** Объект как хранит информацию, так и способен ее преобразовывать. В принципе любой элемент решаемой задачи (дом, собака, услуга, химическая реакция, город, космический корабль и т. д.) может представлять собой объект. Объект можно представить себе как швейцарский нож: он является набором различных ножей и "открывашек" (хранение), но в то же самое время им мы можем резать или открывать что-либо (преобразование).
2. **Программа — совокупность объектов, указывающих друг другу что делать.** Для обращения к одному объекту другой объект "посылает ему сообщение". Как вариант возможно и "ответное сообщение". Программу можно представить себе как совокупность к примеру 3 объектов: писателя, ручки и листа бумаги. Писатель "посылает сообщение" ручке, которая в свою очередь "посылает сообщение" листу бумаги — в результате мы видим текст (посыл сообщения от листа к писателю).
3. **Каждый объект имеет свою собственную "память" состоящую из других объектов**. Таким образом, программист может скрыть сложность программы за довольно простыми объектами. К примеру, дом (достаточно сложный объект) состоит из дверей, комнат, окон, проводки и отопления. Дверь, в свою очередь, может состоять из собственно двери, ручки, замка и петель.
4. **У каждого объекта есть тип**. Иногда тип называют еще и **классом**. Класс (тип) определяет какие сообщения объекты могут посылать друг другу.
5. **Все объекты одного типа могут получать одинаковые сообщения.** К примеру у нас есть 2 объекта: синяя и красная кружки. Обе разные по форме и материалу. Но из обеих мы можем пить (или не пить, если они пустые). В данном случае кружка — это тип объекта.



Самое лаконичное описание объекта предложил Гради Буч (*Grady Booch - американский инженер, руководитель исследований в IBM Research*): "Объект обладает состоянием, поведением и индивидуальностью".

Классы объектов и отдельные объекты.

Отдельно взятый объект представляет один определенный документ, форму, ячейку рабочего листа или другой организованный элемент информации. **Класс** - это тип данных, а **объект** - экземпляр типа **класс**. "Кружка" - это класс (тип). А уж которая, - синяя или красная, - это два разных объекта (экземпляра), типа "кружка". **Класс** описывает переменные, свойства, процедуры и события **объекта.**

Объекты в VBA состоят в определённой **иерархии**. Помимо того, что каждый из них обладает собственными свойствами, методами и событиями, объект, возглавляющий иерархию, служит контейнером для одного или нескольких других. Эти объекты, в свою очередь, содержат другие объекты и т.д.

Объектная модель

Программирование на VBA можно рассматривать, как управление объектами приложения. Объектом является почти все - таблицы, книги, ячейки, диапазоны ячеек, даже сам Excel. У объектов есть свойства (например, ширина, высота, цвет) и методы (такие как Select например). Доступ к свойствам и методам осуществляется через точку. Например так: Cells(1,1).Select - выделить первую ячейку первой страки в текущей таблице. Объекты могут быть вложенными один в другой. Ячейка вложена в таблицу, таблица в книгу, книга в окно Excel. На VBA это выглядит вот так:

Application.ActiveWorkbook.ActiveSheet.Cells(1,1) = 123

Записываем значение 123 в первую ячейку первой строки текущей таблицы.

В этом случае иерархия выглядит так:

Application
Workbook
.....
Worksheets
.....
Cell
.....

Задача программиста на VBA сводится к изменению свойств объектов и вызову их методов.

Контрольные понятия для изучения.

1. Объект, как основное понятие VBA.
2. Инкапсуляция объекта.
3. Основные черты объектно-ориентированного языка программирования
4. Классы (типы) объектов.

5. Объектная модель программы.

Порядок выполнения.

1. Изучить теоретическую часть и занести в протокол основные положения.
2. Для студентов с номером по списку кратному трём, создать в Microsoft Word процедуру возведения числа в квадрат методом суммирования нечётных чисел (метод Пифагора) с применением диалоговых окон **InputBox** и **MsgBox.**
3. Для студентов с номером по списку кратному четырём (кроме номеров кратных трём), создать в Microsoft Word процедуру вычисления чисел Фибоначчии в заданном диапазоне с применением диалоговых окон **InputBox** и **MsgBox**.
4. Оставшимся студентам с нечётными номерами создать в Microsoft Word процедуру вычисления числа Фибоначчи по его порядковому номеру с применением диалоговых окон **InputBox** и **MsgBox**.
5. Остальным студентам создать в Microsoft Word процедуру вычисления числа π (пи) с применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do Until … Loop**.
6. Отладить и запустить эти программы на выполнение в среде Microsoft Word..
7. Описать в комментариях объекты и классы объектов, использованных в программах. Занести текст созданных процедур в протокол.
8. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №9

Особенности применения VBA в Microsoft Excel.

Цель: изучить основные приёмы создания процедур в Microsoft Excel.

Основные вопросы.

1. Объекты OLE и ActiveX.
2. Объекты. Методы и свойства объектов.
3. Классы.
4. Иерархия объектов.
5. Работа с методами и свойствами объектов в среде Microsoft Excel.

Семейство (объект Collection) представляет собой объект, содержащий несколько других объектов, как правило одного типа. Например, объект WorkBooks (Рабочие книги) содержит все открытые объекты WorkBook (Рабочая книга). Каждый элемент семейства нумеруется и может быть идентифицирован либо по номеру, либо по имени. Например, worksheets (1) обозначает первый рабочий лист активной книги, а worksheets (Лист1) - рабочий лист с именем Лист1.

Объекты OLE и ActiveX.

В VBA используется механизм **OLE** (**Object Linking and Embedding** - связывание и внедрение объектов), который позволяет взаимодействовать с любыми программами, поддерживающими **OLE**. Примером элементов, которые можно интегрировать при помощи механизма **OLE**, являются вставляемые объекты **OLEObject**, создаваемые, например, при помощи программ.

Объекты. Методы и свойства объектов.

WordArt, ClipArt и т. д. Все **OLE-объекты** рабочего листа образуют семейство **OLEObjects**. Вручную в рабочий лист **OLE-объекты** вставляются командой **Вставка/Объект** с выбором в появившемся диалоговом окне Вставка объекта из списка на вкладке Создание внедряемого объекта. **OLE-объект** отличается от обычного тем, что при выборе внедренного объекта (при перемещении на него указателя и щелчке кнопкой мыши) активизируется программа, связанная с этим объектом, и меню приложения заменяется меню программы, его создавшей. Теперь можно, не выходя из основного приложения, работать с данным объектом,

редактируя и видоизменяя его средствами создавшей его программы. Кроме того, **OLE**-технология обладает так называемым свойством **Automation**, с помощью которого можно устанавливать свойства, применять методы и обрабатывать события внедренных объектов как обычных объектов приложения.

С 1996 г. фирма Microsoft ввела новую терминологию, и теперь то, что раньше именовалось **OLE-объектом**, называется объектом **ActiveX**, а **OLE Automation** называется **ActiveX Automation**.

Классы

Важнейшим понятием VBA является **класс**. **Класс** обычно описывается как проект, на основе которого впоследствии будет создан конкретный объект. Таким образом, класс определяет имя объекта, его свойства и действия, выполняемые над объектом. В свою очередь, каждый объект, в соответствии с описанным выше, является экземпляром класса.

Иерархия объектов. Ссылки на объект.

Объектная библиотека VBA содержит более 100 различных объектов, находящихся на разных уровнях иерархии. **Иерархия** определяет связь между объектами и показывает пути доступа к ним.

Полная ссылка на объект состоит из ряда имен вложенных последовательно друг в друга объектов. Разделителями имен объектов в этом ряду являются точки, ряд начинается с объекта **Application** и заканчивается именем самого объекта.

Например, полная ссылка на ячейку **A1** рабочего листа **Лист1** рабочей книги с именем **Кафедра** имеет вид:

Application.Workbooks("Кафедра" heets("Листі").Range("Al")

Приводить каждый раз полную ссылку на объект не обязательно. Обычно достаточно ограничиться только неявной ссылкой на объект. В неявной ссылке, в отличие от полной, объекты, которые активны в данный момент, как правило, можно опускать. В рассмотренном случае, если ссылка на ячейку **A1** дана в программе, выполняемой в среде Excel, то ссылка на объект **Application** может быть опущена, т. е. достаточно привести относительную ссылку:

Workbooks("Кафедра").Worksheets("Лист1").Range("A1")

Если рабочая книга **Кафедра** является активной, то ссылку можно записать еще короче:

Worksheets("Лист1").Range("A1")

Если и рабочий лист **Лист1** активен, то в относительной ссылке вполне достаточно ограничиться упоминанием только диапазона **A1**:

Range("A1")

Методы.

Объект сам по себе не представляет большого значения. Намного значительнее то, какие действия можно совершать над объектом и какими свойствами он обладает. Метод как раз и представляет собой действие, выполняемое над объектом.

Синтаксис применения метода:

Объект.Метод

Например, при помощи метода **Quit** (Закрыть) закрывается приложение (объект **Application**):

Application.Quit

Метод можно применять ко всем объектам семейства. Например, к семейству **chartobjects** (Диаграммы) рабочего листа **Лист1** применен метод **Delete** (Удалить), который приводит к удалению всех диаграмм с рабочего листа **Лист1**:

Worksheets("Лист!") .ChartObjects.Delete

Свойства.

Свойство представляет собой атрибут объекта, определяющий его характеристики, такие, как размер, цвет, положение на экране и состояние объекта, например доступность или видимость. Чтобы изменить характеристики объекта, надо просто изменить значения его свойств.

Синтаксис установки значения свойства:

Объект.Свойство = ЗначениеСвойства

В следующем примере изменяется заголовок окна Excel посредством задания свойства **Caption** объекту **Application**:

Application.Caption = "База данных"

Свойство можно изменять сразу у всех объектов семейства. Например, с помощью установки свойству **Visible** (Видимость) значения **False** (Ложь) все рабочие листы активной книги (семейство объектов **Worksheets**) скрываются:

Worksheets.Visible = False

Среди свойств особое место занимают свойства, возвращающие объект.

ActiveWindow	Возвращает активное окно Excel
ActiveWorkbook	Возвращает активную рабочую книгу окна Excel
ActiveSheet	Возвращает активный лист активной рабочей книги
ActiveDialog	Возвращает активное диалоговое окно активного рабочего листа
ActiveChart	Возвращает активную диаграмму активного рабочего листа
ActiveCell	Возвращает активную ячейку активного рабочего листа

Запутаться в свойствах и методах несложно: их существует несколько тысяч. Все объекты обладают свойствами. Например, объект **Range** обладает свойством с названием **Value** . Можно создать оператор VBA , чтобы отобразить свойство **Value** или задать свойству **Value** определенное значение. Ниже приведена процедура, использующая функцию **VBA MsgBox** для отображения окна, в котором представлено значение ячейки **A1** листа **Лист1** активной рабочей книги.

Sub ShowValue()

MsgBox Worksheets("Лист1").Range("Al").Value

End Sub

Ниже приведена процедура по изменению значения ячейки **A1** путем определения значения свойства **Value** .

Sub ChangeValue()

Worksheets("Лист1").Range("Al").Value = 123

End Sub

После выполнения этой процедуры ячейка **Al** листа **Лист1** получает значение **123.** Многие объекты имеют свойство по умолчанию. Для объекта **Range** свойством по-умолчанию является **Value**. Следовательно, выражение **Value** в приведенном выше коде можно опустить, и ничего не изменится. Однако лучше включать ссылку на свойство, даже если оно используется по умолчанию.

Кроме свойств, объекты характеризуются методами. Метод – это действие, которое выполняется над объектом. Ниже приведен простой пример использования

метода **Clear** по отношению к диапазону ячеек. После выполнения этой процедуры ячейки **A1: C3** листа **Лист1** станут пустыми, и дополнительное форматирование ячеек будет удалено.

```
Sub ZapRange()
Worksheets("Лист1").Range("A1:C3").Clear
End Sub
```

Если необходимо удалить значения в диапазоне, но оставить форматирование, используйте метод **ClearContents** объекта **Range** .

Многие методы получают аргументы, определяющие выполняемые над объектом действия более детально. Далее приводится пример, в котором ячейка **A1** копируется в ячейку **B1** с помощью метода **Copy** объекта **Range**. В данном примере метод **Copy** получает один аргумент (адрес ячейки, в которую следует скопировать данные). Обратите внимание, что в примере используется символ продолжения строки (пробел и подчеркивание).

```
Sub CopyOne()
Worksheets ("Лист1") .Range("A1") .Copy _
Worksheets("Лист 1 ").Range ( "B1")
End Sub
```

Можно не применять этого символа, а ввести оператор в одну строку следующим образом:

```
Sub CopyOne()
Worksheets ("Лист1") .Range("A1") .Copy Worksheets("Лист 1 ").Range ( "B1")
End Sub
```

В среде программистов VBA определение аргументов методов и свойств часто вызывает определенные трудности. Некоторые методы используют аргументы для дальнейшего уточнения действия; отдельные свойства используют аргументы для дальнейшего определения значения свойства. Иногда один или несколько аргументов вообще применять не обязательно.

Если метод использует аргументы, они указываются после названия метода и разделяются запятыми. Если метод использует необязательные аргументы, то можете пропустить их, оставив пустые места. Рассмотрим метод **protect** объекта рабочей книги. В справочной системе дается информация о том, что метод **Protect** имеет три аргумента: **пароль, структура, окна**. Эти аргументы соответствуют параметрам в

диалоговом окне **Защита книги**. К примеру, если требуется защитить рабочую книгу под названием **MyBook . xls** , используется такой оператор:

Workbooks("MyBook.xls").Protect "xyzzy", True, False

В данном случае рабочая книга защищена паролем (первый аргумент). Также защищена структура рабочей книги (второй аргумент), но не ее окна (третий аргумент).

Если вы не хотите присваивать пароль, можно применить такой оператор:

Workbooks("MyBook.xls").Protect , True, False

Обратите внимание, что первый аргумент пропущен, а его место обозначено с помощью запятой.

Существует и другой подход (причем в этом случае программу удобнее будет читать) – использование именованных аргументов. Применим именованные аргументы для предыдущего примера.

Workbooks("MyBook.xls").Protect Structure:=True, _

Windows:=False

Использование именованных аргументов – хорошая идея, особенно в методах с большим количеством необязательных аргументов, когда следует использовать только некоторые из них. При использовании именованных аргументов не требуется оставлять место для пропу-щенных аргументов.

Методы VBA

Activate()	выделяет текущий диапазон и устанавливает курсор ввода на его первую ячейку.
AddComment()	возможность добавить комментарий к ячейке. Ячейка будет помечена красным уголком, а текст комментария будет показан в виде всплывающей подсказки. Этот метод можно вызвать только для диапазона, состоящего из одной ячейки. То же самое на графическом экране можно сделать при помощи меню **Вставка -> Примечание.**

AutoFill()	возможность использовать автозаполнение для диапазона (например, если первые две ячейки будут заполнены как 1 и 2, то дальше в автоматическом режиме будет продолжено: 3, 4, 5 и т.п.)
AutoFit()	автоматически поменять ширину всех столбцов и высоту всех строк в диапазоне, чтобы туда уместился текст ячеек. Можно применять только к тем диапазонам, которые состоят из набора столбцов (полностью) или набора ячеек (также полностью), иначе будет ошибка.
AutoFormat()	возможности использовать один из стилей автоформатирования (то, что на графическом экране доступно через меню **Формат -> Автоформат**).
BorderAround()	возможность поместить диапазон в рамку с выбранными вами параметрами.
Clear…	позволяют очистить содержимое диапазона от значений, форматирования, комментарий и т.п.
Consolidate()	возможность слить данные нескольких диапазонов (в том числе на разных листах) в один диапазон, используя при этом выбранную вами агрегатную функцию.
Copy()	возможность скопировать диапазон в другое место. Если место назначения не указано, он копируется в буфер обмена. Аналогично работает метод *Cut()*, при котором данные исходного диапазона вырезаются.
CopyFromRecordset()	очень удобный метод, который позволяет вставить данные из объекта ADO Recordset на лист Excel, начиная с верхнего левого угла указанного диапазона.
DataSeries()	метод, который может сэкономить множество времени и избежать возни с функциями даты и времени. Этот метод позволяет увеличить вами значения даты в диапазоне на указанный вами временной интервал. Например, если у вас в диапазоне стоит первое января, то при помощи этого метода можно сгенерировать первое число любого другого месяца.

Delete()	удаляет данные текущего диапазона. В качестве необязательно параметра можно определить, с какой стороны будут сдвигаться ячейки на место удаленных.
Dirty()	пометить ячейки диапазона как "грязные". Такие ячейки будут пересчитаны при следующем же пересчете. Обычно используется, когда Word сам не может догадаться, что их нужно пересчитать. Пересчитать ячейки диапазона можно и принудительно - при помощи метода *Calculate()*.
методы *Fill…* (*FillDown()*, *FillUp()*, *FillLeft()*, *FillRight()*)	позволяют размножить одно и то же значение по ячейкам диапазона в указанном вами направлении.
Find()	позволяет произвести поиск по ячейкам диапазона и вернуть новый объект Range, который представляет первую ячейку, в котором было найдено нужное значение. У этого метода есть множество необязательных параметров, которые позволяют определить направление поиска, чувствительность к регистру, искать все значение ячейки или часть и т.п. Методы FindNext() и FindPrevious() позволяют продолжить поиск, начатый методом Find(), в разных направлениях.
GoalSeek()	позволяет применить автоподбор значений для функции Excel программным способом. На графическом экране то же самое можно сделать при помощи меню **Сервис** -> **Подбор параметра**.
Insert()	позволяет вставить ячейки в диапазон, сдвинув остальные (вы можете выбрать - вправо или вниз).
Justify()	позволяет равномерно распределить текст по диапазону. Если в данный диапазон он не помещается, он будет распространен на соседние ячейки (с перезаписью их значений).

Merge()	позволяет слить все ячейки диапазона в одну. При этом останется только одно значение - верхней левой ячейки. Разбить обратно такую слитую ячейку на несколько обычных можно при помощи метода *UnMerge()*.
Parse()	позволяет разбить одну ячейку на несколько по указанному вами шаблону (например, чтобы отделить код города от номера телефона).
PasteSpecial()	операция, дополняющая Copy() и Cut(). Она позволяет вставить то, что лежит в буфере обмена, с указанием специальных параметров вставки (вставлять с добавлением к существующим данным, с умножением, вычитанием, делением и т.п.)
PrintOut() **и** ***PrintPreview()***	позволяют вывести диапазон на печать или открыть режим просмотра перед печатью..
Replace()	метод, дополняющий метод Find(). Позволяет проводить поиск и замену значений в диапазоне.
Select()	возможность выделить указанный диапазон. Объекта Selection в Excel нет - вместо него есть возможность получить объект Range, представляющий выделенную область.
Show()	экран будет проскроллирован таким образом, чтобы показать указанный диапазон.
ShowDependents()	показать стрелки для ячеек, которые зависят от указанного диапазона (только первый уровень зависимости) или эти стрелки убрать. Обратный метод - *ShowPrecedents()*.
ShowErrors()	показать источник ошибки для указанной ячейки.
Sort()	возможность произвести сортировку ячеек в диапазоне. Можно использовать большое количество необязательных параметров для настройки сортировки.
SortSpecial()	с учетом особенностей азиатских языков.

Speak()	удивительный метод, который позволяет зачитывать вслух содержимое диапазона (можно определить, в каком направлении и будут ли зачитываться формулы). К сожалению, в локализованной версии Excel не работает.
SpecialCells()	очень удобный метод, который позволяет вернуть объект Range, включающий в себя все ячейки определенного типа (пустые, с ошибками, с комментариями, последние, с константами, с формулами, с определенным форматированием) и с определенным значением. Например, чтобы вернуть объект Range, состоящий из всех пустых ячеек диапазона, можно использовать код
SubTotal()	позволяет посчитать итоговое значение для диапазона (можно выбрать агрегатную функцию и множество других параметров).
Table()	позволяет создать таблицу на основе передаваемого столбца, строки и функции, которую нужно использовать для вычисления ячеек таблицы. Пример из документации по этому методу позволяет автоматически сгенерировать таблицу умножения.
TextToColumns()	сложный метод, который позволяет разбить столбцы в диапазоне на несколько столбцов в соответствии с определенным алгоритмом. Принимает множество необязательных параметров.

Пример программы работы с ячейкой текущей книги Excel.

```
Sub Проба_1()
'Открывает форму на ограниченное время

  Sheets("Лист1").Select 'Переход на Лист1
  MsgBox ("После нажатия кнопки" & Chr(10) & "форма будет показана через 5 секунд!") 'Окно сообщения
  Application.Wait (Now + TimeValue("0:00:5")) 'На 5 секунд приложение деактивировано
  UserForm1.Show 'Показывает форму UserForm1

  Range("C5").Activate 'Активирует ячейку C5
  Dim Box_C5 As Range 'Объявляет переменную как диапазон ячеек
  Set Box_C5 = Worksheets("Лист1").Range("C5") 'Устанавливает переменную как ячейку C5
```

```
  Box_C5.Value = "Мое значение" 'Меняет значение в ячейке
  ActiveCell.Interior.ColorIndex = 36 'Меняет цвет ячейки
  MsgBox ("Надпись выведена" & Chr(10) & "Цвет ячейки изменён") 'Окно сообщения
  ActiveCell.Interior.ColorIndex = 26 'Меняет цвет ячейки
  MsgBox ("Меняем цвет ячейки ещё раз") 'Окно сообщения
  Box_C5.Clear 'Очищает ячейку полностью
  MsgBox ("Ячейка очищена") 'Окно сообщения
End Sub
```

Контрольные понятия для изучения.

1. Объекты OLE и ActiveX.
2. Классы и иерархия объектов.
3. Методы и свойства объектов.
4. Работа с аргументами методов и свойств объектов в среде Microsoft Excel.

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

1. Для студентов с номером по списку кратному трём, написать процедуру создания нового листа в книге Microsoft Excel, а затем его удаления с применением диалоговых окон **InputBox** и **MsgBox.**
2. Для студентов с номером по списку кратному четырём (кроме номеров кратных трём), создать процедуру подсчёта листов в книге Microsoft Excel с применением диалоговых окон **InputBox** и **MsgBox**.
3. Оставшимся студентам с нечётными номерами создать процедуру, копирующую столбец в книге Microsoft Excel с одного листа на другой и затем его очищающий с применением диалоговых окон **InputBox** и **MsgBox**.
4. Остальным студентам создать процедуру в Microsoft Excel копирования диапазона строк с одного листа книги на другой с применением диалоговых окон **InputBox** и **MsgBox** и использованием структуры **Do Until … Loop**.
5. Отладить и запустить эти программы на выполнение в среде Microsoft Excel.
6. Описать в комментариях объекты и действия над ними использованные в программах. Занести текст созданных процедур в протокол.
7. Отобразить в протоколе основные трудности, возникшие у Вас при создании программы.

ЛАБОРАТОРНАЯ РАБОТА №10

Численные методы. Регрессионный анализ. Приближение функций по методу наименьших квадратов.

Цель: научиться определять параметры функций с помощью метода наименьших квадратов.

Основные вопросы.

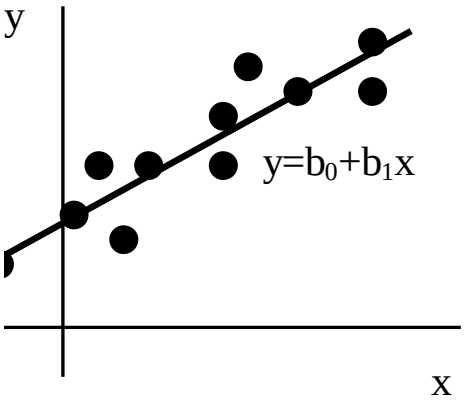
1. Линейный регрессионный анализ.
2. Вычисление параметров линейной функции с помощью МНК.
3. Среднеквадратичная погрешность.
4. Коэффициент корреляции.
5. Преобразования, сводящие нелинейную регрессию к линейной.

Регрессионный анализ

(приближение функций по методу наименьших квадратов)

Линейный парный регрессионный анализ заключается в определении параметров эмпирической линейной зависимости

$$\mathbf{y(x)=b_0+b_1x}$$



описывающей связь между некоторым числом **N** пар значений $\mathbf{x_i}$ и $\mathbf{y_i}$, обеспечивая при этом наименьшую среднеквадратичную погрешность. Графически эту задачу можно представить следующим образом – в облаке точек $\mathbf{x_iy_i}$ требуется провести прямую так, чтобы величина всех отклонений отвечала условию:

$$U = \sum_{i=1}^{N}\left[y_i - y\left(x_i\right)\right]^2 \Rightarrow \min$$

где $\mathbf{y(x_i)}$ – зависимость $\mathbf{y(x)=b_0+b_1x}$. Для этого нужно приравнять нулю частные производные

$$\frac{\partial U}{\partial b_0} = \sum_{i=1}^{N}\left[y_i - \left(b_0 + b_1x_i\right)\right], \quad \frac{\partial U}{\partial b_1} = \sum_{i=1}^{N}\left[y_i - \left(b_0 + b_1x_i\right)x_i\right],$$

что даёт для определения не известных коэффициентов $\mathbf{b_0}$ и $\mathbf{b_1}$ систему линейных уравнений

$$\begin{cases} b_0 N + b_1 \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} y_i \\ b_0 \sum_{i=1}^{N} x_i + b_1 \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} x_i y_i \end{cases}$$

Решение этой системы:

$$b_0 = \frac{\sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i y_i}{N \cdot \sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2};$$

$$b_1 = \frac{N \cdot \sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N \cdot \sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2}.$$

при этом среднеквадратичная погрешность:

$$\sigma^2 = \frac{1}{N} \left[\sum_{i=1}^{N} y_i^2 - b_1 \sum_{i=1}^{N} y_i - b_0 \sum_{i=1}^{N} x_i y_i \right].$$

Коэффициент линейной парной корреляции:

$$R = \frac{\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i \right) / N}{\sqrt{\sum_{i=1}^{N} x_i^2 - \frac{\left(\sum_{i=1}^{N} x_i \right)^2}{N}} \sqrt{\sum_{i=1}^{N} y_i^2 - \frac{\left(\sum_{i=1}^{N} y_i \right)^2}{N}}}$$

Коэффициент парной корреляции характеризует степень отклонения связи между $\mathbf{x_i}$ и $\mathbf{y_i}$ от прямой. Если **R** близок к **1**, то эта связь линейна, т. е. $\mathbf{y_i = b_0 + b_1 x}$, причём знак **R** определяет знак коэффициента $\mathbf{b_0}$.

Преобразования сводящие нелинейную регрессию к линейной

(её параметры помечены штрихами)

№	Функция y(x)	x’	y’	b_0	b_1
1	b_0+b_1x	x	y	b'_0	b'_1
2	$1/(b_0+b_1x)$	x	1/y	b'_0	b'_1
3	b_0+b_1/x	1/x	y	b'_0	b'_1
4	$x/(b_0+b_1x)$	x	x/y	b'_0	b'_1
5	$b_0 b_1^x$	x	lg y	$10^{b'_0}$	$10^{b'_1}$
6	$b_0 e^{(b1.x)}$	x	ln y	$e^{b'0}$	b'_1
7	$b_0 10^{b1.x}$	x	lg y	$10^{b'0}$	b'_1

№	Функция y(x)	x’	y’	b_0	b_1
8	$1/(b_0+b_1e^{-x})$	e^{-x}	1/y	b'_0	b'_1
9	$b_0x^{b1.x}$	lg x	lg y	$10^{b'0}$	b'_1
10	b_0+b_1lgx	lg x	y	b'_0	b'_1
11	b_0+b_1lnx	ln x	y	b'_0	b'_1
12	$b_0/(b_1+x)$	x	1/y	$1/b'_0$	b'_1
13	$b_0x/(b_1+x)$	1/x	1/y	$1/b'_0$	b'_0/b'_1
14	$b_0e^{(b1/x)}$	1/x	ln y	$e^{b'0}$	b'_0/b'_1
15	$b_0\cdot10^{b1/x}$	1/x	lg y	$10^{b'0}$	b'_1
16	$b_0+b_1x^n$	x^n	y	b'_0	b'_1

Контрольные понятия для изучения.

1. Алгоритм вычисления коэффициентов линейной функции с помощью МНК.
2. Вычисление и физический смысл среднеквадратичной погрешности.
3. Вычисление и физический смысл коэффициента корреляции.
4. Принципы преобразований, сводящих нелинейную регрессию к линейной.

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В редакторе электронных таблиц "Excel" создать таблицу произвольных исходных данных и вычислить:

- Коэффициенты линейной регрессии по методу наименьших квадратов.
- Среднеквадратичную погрешность вычислений.
- Коэффициент парной корреляции.
- Выполнить линеаризацию исходных данных по 16 зависимостям приведенным в таблице.

Поместить на лист таблицы исходные данные и результаты вычислений.

ЛАБОРАТОРНАЯ РАБОТА №11

Численные методы. Интерполяция и экстраполяция.

Цель: научиться определять неизвестные значения функций с помощью интерполяционных методов.

Основные вопросы.

1. Понятие интерполяции и экстраполяции. Интерполяционный полином
2. Обратная интерполяция.
3. Многоинтервальная интерполяция.
4. Интерполяция полиномом Лагранжа.
5. Интерполяция полиномом Ньютона.

Понятие интерполяции и экстраполяции

В вычислительной математике существенную роль играет интерполяция функций, т.е. построение по заданной функции другой (как правило, более простой), значения которой совпадают со значениями заданной функции в некотором числе точек. На практике часто возникает задача о восстановлении непрерывной функции по ее табличным значениям, например полученным в ходе некоторого эксперимента.

Интерполяция функции **y(x)** одной переменной **x**, заданной **(n+1)** узлами $\mathbf{y_i(x_i)}$, где **i=0,1,2,...n**, заключается в нахождении значений **y** по значениям **x**, находящимися в промежутках между узлами $\mathbf{x_i}$. При интерполяции функция **y(x)** заменяется интерполяционным полиномом **P(x)**, значения которого $\mathbf{P_i(x_i)}$ в узлах точно совпадают с $\mathbf{y_i(x_i)}$. Значение **n** задаёт степень полинома **P(x)**.

Экстраполяция – получение значений **y(x)** при **x**, не принадлежащему отрезку $\mathbf{[x_0, x_n]}$ или $\mathbf{[x_0, x_{n+1}]}$.

Обратная интерполяция – процесс нахождения значений **x** по значениям **y**.

Многоинтервальная интерполяция – заключается в интерполяции $\mathbf{y_i(x_i)}$ в ряде частных интервалов (ограниченных двумя узлами или группой узлов) отдельными полиномами невысокой степени.

Интерполяция полиномом Лагранжа

Интерполяция полиномом Лагранжа при произвольном расположении узлов в общем случае сводится к вычислению **y(x)=Ln(x)** с помощью интерполяционного полинома имеющего вид:

$$L_n = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} y_0 +$$
$$+ \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} y_1 + \ldots$$
$$+ \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})} y_n$$

Легко видеть, что при подстановке в формулу значения $\mathbf{x = x_i}$ все члены суммы, кроме **i**-го, обращаются в нуль (в числителе дробей, входящих в эти члены, появляются множители, равные нулю), а в **i**-м члене числитель становится равным знаменателю и дробь обращается в единицу, так что остается лишь множитель $\mathbf{y_i}$, т.е. получается, что $\mathbf{L_i = y_i}$. Таким образом, интерполяционный многочлен Лагранжа совпадает в заданных точках с заданными значениями неизвестной функции.

Алгоритм интерполяции полиномом Лагранжа на языке Q-Basic.

```
        Cls
10      Print "Интерполяция по Лагранжу для N+1 узлов"
        ?:?
              DefInt I,J,N
              DefDbl A,B,C,D,S
20      Input "Введите число узлов N="N
              Dim A(N),B(N)
30      For I=0 To N
              Print "Введите X"I
                    Input A(I)
              Print "Введите Y"I
                    Input B(I)
50      Next I
60      Input "Введите X="X
        S=0
70            If X=A(0) Then
                    Print "Y(X)="B(0)
                    GoTo 60
              End If
80      For J=1 To N
              C=1
              For I=1 To N
90                  D=A(J)-A(I)
                    If I=J Then D=X-A(j)
100                 If D=0 Then
                          Print "Y(X)="B(I)
                          GoTo 60
```

```
                    End If
110                 C=C*(X-A(I))/D
                Next I
120             S=S+C*B(J)
        Next J
130     Print "Y("X")="S
        GoTo 60
        End
```

Алгоритм интерполяции полиномом Лагранжа на VBA.

```
Sub Интерполяция_Лагранжа()

Const N = 4 'Количество точек интерполяции
Dim a As Double, b As Double, mx(N) As Double
Dim my(N) As Double, Row As Integer, i As Integer
Dim x As Double, hp As Double
a = 0: b = 1
Sheets("Лист1").Select
Call - прочитатьТочкиИнтерполяции(N, mx, my)
hp = 0.05 'Шаг между двумя точками для построения графика функции
Row = 1: Cells(Row, 3) = "x": Cells(Row, 4) = "L(x)"
'Цикл по точкам для построения графика функции y=f(x)
For x = a To b + hp / 10 Step hp
Row = Row + 1
Cells(Row, 3) = x
Cells(Row, 4) = Лагранж(N, x, mx, my) 'Вычисление LN(x)
Next x
Call GraphicsL(Row, N, 1)
End Sub

'Подпрограмма, строящая график функции и точки интерполяции
'nGr - количество графиков
Sub GraphicsL(Row As Integer, N As Integer, Ngr As Integer)
Dim s As String, alph As String
alph = "DEFGHIJKLMNOPQ"
'Область данных для графика функции
s = "C1:" + Mid(alph, Ngr, 1) + Trim(Str(Row))
Charts.Add
ActiveChart.ChartType = xlXYScatterLinesNoMarkers
ActiveChart.SetSourceData
Source:=Sheets("Лист1").Range(s), PlotBy:=xlColumns
ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист1"
'Изображение точек интерполяции (Добавить второй график)
ActiveChart.SeriesCollection(Ngr).Select
ActiveChart.SeriesCollection.NewSeries
'Массив абсцисс точек интерполяции
s = "=Лист1!R2C1:R" + Trim(Str(N + 2)) + "C1"
ActiveChart.SeriesCollection(Ngr + 1).XValues = s
'Массив ординат точек интерполяции
s = "=Лист1!R2C2:R" + Trim(Str(N + 2)) + "C2"
ActiveChart.SeriesCollection(Ngr + 1).Values = s
'Название кривой (легенда)
ActiveChart.SeriesCollection(Ngr + 1).Name = "=Лист1!R1C2"
ActiveChart.SeriesCollection(Ngr + 1).Select
'Тип графика для точек интерполяции
ActiveChart.SeriesCollection(Ngr + 1).ChartType = xlXYScatter

End Sub
```

Интерполяция полиномом Ньютона

Форма Лагранжа записи интерполяционного полинома не является единственно возможной. Кроме этого, она имеет определенные неудобства. Так предположим, задан набора точек $(x_k, y_k)_{k=1,2,\ldots,n+1}$ и по нему построен интерполяционный полином в форме Лагранжа

$$L_n(x) = \sum_{k=1}^{n+1} y_k \frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{k-1})\cdot(x-x_{k+1})\cdot\ldots\cdot(x-x_n)\cdot(x-x_{n+1})}{(x_k-x_1)\cdot(x_k-x_2)\cdot\ldots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\ldots\cdot(x_k-x_n)\cdot(x_k-x_{n+1})}.$$

Тогда очевидно, что при добавлении новой точки к исходному набору точек придется заново конструировать интерполяционный полином. Этого недостатка лишен интерполяционный полином, записанный в рассмотренной ниже форме Ньютона.

Важно понимать, что каким бы способом не строился интерполяционный полином по заданному набору точек, результат всегда получается один и тот же (с точностью до ошибок, возникающих при округлении вещественных чисел), поскольку через заданные **n+1** точки проходит ровно один полином **n** -ой степени

Для вывода интерполяционного полинома в форме Ньютона понадобится понятие о разделенных разностях.

Разделенные разности и некоторые их свойства.

Пусть нам заданы различные точки $x_1, x_2, \ldots, x_{n+1}$ и функция *f*(x). Первой разделенной разностью функции $f(x)$ в точках x_i, x_{i+1}называется величина

$$[x_i, x_{i+1}]f = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}.$$

Разделенные разности более высоких порядков определяются рекуррентно. Например разделенная разность второго порядка в точках x_1, x_2, x_3определяется через две разделенные разности первого порядка $[x_1, x_2]f$ и $[x_2, x_3]f$ следующим образом:

$$[x_1, x_2, x_3]f = \frac{[x_2, x_3]f - [x_1, x_2]f}{x_3 - x_1},$$

Разделенная разность второго порядка в точках x_1, x_2, x_3определяется так:

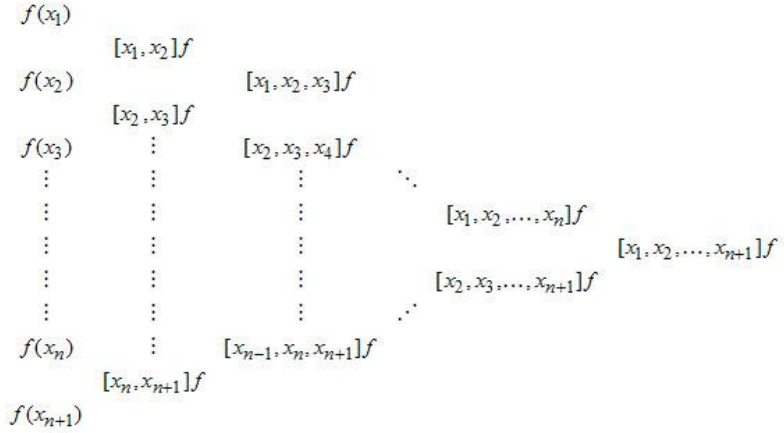
$$[x_2, x_3, x_4]f = \frac{[x_3, x_4]f - [x_2, x_3]f}{x_4 - x_2}$$

и т.д.

В общем случае разделенная разность s-го порядка определяется через две разделенные разности $(s-1)$-го порядков

$$[x_i, x_{i+1}, \ldots, x_{i+s}]f = \frac{[x_{i+1}, x_{i+2}, \ldots, x_{i+s}]f - [x_i, x_{i+1}, \ldots, x_{i+s-1}]f}{x_{i+s} - x_i}.$$

Разделенные разности удобно представить в виде треугольной таблицы



Получение интерполяционного полинома в форме Ньютона

Интерполяция с одноимённым получением коэффициентов полинома

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

может выполняться с применением интерполяционной формулы Ньютона:

$$P_{n-1}(x) = y_1 + (x - x_1) f(x_1; x_2) + (x - x_1)(x - x_2) f(x_1; x_2; x_3) + \\ + (x - x_1)(x - x_2) \ldots (x - x_{n-1}) f(x_1; x_2; \ldots; x_n)$$

Эту формулу можно также записать в следующем более компактном виде:

$$L_n(x) = \sum_{i=0}^{n} b_i (x - x_1)(x - x_2) \ldots (x - x_i),$$

где:

$$b_0 = f(x_1), \quad b_1 = [x_1, x_2]f, \quad b_2 = [x_1, x_2, x_3]f, \ldots, \quad b_n = [x_1, x_2, \ldots, x_{n+1}]f.$$

Мы можем считать, что точки пронумерованы в обратном порядке, тогда

$$L_n(x) = \sum_{i=1}^{n+1} d_i (x - x_{n+1})(x - x_n)\dots(x - x_{i+1}),$$

где

$$d_{n+1} = f(x_{n+1}),\ d_n = [x_n, x_{n+1}]f,\ d_{n-1} = [x_{n-1}, x_n, x_{n+1}]f,\ d_1 = [x_1, x_2, \dots, x_{n+1}]f$$

Тогда при вычислении значения интерполяционного полинома в форме Ньютона требуется найти разделенные разности, стоящие внизу треугольной таблицы и вычислить значение приведенного выше полинома $L_n(x)$ от заданного значения ***x***.

Алгоритм интерполяции полиномом Ньютона на языке Q-Basic.

```
Cls
10      Print "Построение интерполяционного полинома Ньютона"
20      Print "и интерполяция при произвольно расположенных узлах"
        DefDbl A,F,P,R,S,W,X,Y
        DefInt i,j,k,l,m,N,v
        ?:?
30      Input "Задайте число узлов N="N
        Dim A(N),F(N),X(N),Y(N)
40      For i=1 To N
            Print "Введите X"i", Y"i
50              Input X(i),Y(i)
        Next i
        A(1)=1
        F(N)=Y(1)
60      For i=1 To N-1
            F(i)=0
        Next I
70      For k=1 To N-1
            For i=1 To N-k
80              Y(i)=(Y(i+1)-Y(i))/(X(i+k)-X(i))
            Next i
90          R=1
            If k/2-Int(k/2)<>0 Then R=-1
100         P=1
            For j=1 To k
                P=P*X(j)
            Next j
110         A(k+1)=R*P
            If k=1 Then 170
120         For l=1 To k
                W=0
                For m=1 To l
130                 R=1
                    If k/2-Int(k/2)<>0 Then R=-1
140                 S=0
                    For v=1 To k
                        S=S+R*(1/X(v))^m
                    Next v
150                 W=W+(-R)*A(k+1+m-l)*S
                Next m
```

```
160                     A(k-l+1)=W/l
                Next l
170             For j=N To N-k Step -1
180                     F(j)=F(j)+A(j-N+k+1)*Y(1)
                Next j
        Next k
190     Print "Коэффициенты степенного многочлена"
200     For i=1 To N
                Print"A"N-1"="F(i)
        Next i
210     Input "Введите значение X="X
        S=F(1)
        For i=1 To N-1
220             S=S*X+F(i+1)
        Next i
230     Print "Y(X)="S
        GoTo 210
        End
```

Контрольные понятия для изучения.

1. Понятие интерполяции и экстраполяции.
2. Обратная интерполяция. Многоинтервальная интерполяция.
3. Интерполяционный полином
4. Интерполяция полиномом Лагранжа.
5. Интерполяция полиномом Ньютона.

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В редакторе электронных таблиц "Excel" создать таблицу произвольных исходных данных и вычислить:

- Значения неизвестных точек таблично заданной функции методом Лагранжа, использовав в качестве примера и отредактировав представленные алгоритмы.
- Значения неизвестных точек таблично заданной функции методом Ньютона, использовав в качестве примера представленный алгоритм на языке Q-Basic.

Поместить на лист таблицы исходные данные и результаты вычислений.

ЛАБОРАТОРНАЯ РАБОТА №12

Численные методы. Решение нелинейных и трансцендентных уравнений.

Цель: практически ознакомиться с численными методами решения нелинейных и трансцендентных уравнений.

Основные вопросы.

1. Понятие трансцендентного уравнения.
2. Метод простых итераций.
3. Метод Ньютона (касательных).
4. Метод дихотомии (деления отрезка пополам).
5. Метод хорд.
6. Метод Секущих.

Понятие трансцендентного уравнения

Нахождение точных корней *алгебраического* или *трансцендентного* уравнения (т.е. уравнения неалгебраического, например, тригонометрического, логарифмического или иррационального) является зачастую достаточно сложной задачей, не решаемой аналитически с помощью конечных формул. Кроме того, иногда на практике уравнение содержит коэффициенты, значения которых заданы приблизительно, так что говорить о точном решении уравнений в таких случаях вообще не имеет смысла. Поэтому задачи *приближенного* определения корней уравнения и соответствующей оценки их точности очень важны.

Рассмотрим уравнение:

$$F(x)=0,$$

где функция $F(x)$ – непрерывна и определена на некотором интервале $a<x<b$. В ряде случаев потребуется существование и непрерывность первой и второй производных этой функции: $F'(x)$ и $F''(x)$, что каждый раз будет оговариваться особо.

Всякое значение ξ, при котором F(x) обращается в нуль:

$$F(\xi) = 0,$$

называется корнем уравнения (1) или нулем функции F(x).

Будем считать, что уравнение имеет только изолированные корни, т.е. для каждого корня уравнения существует окрестность, не содержащая других корней этого уравнения. Другими словами, на рассматриваемом участке существует только один корень уравнения.

Приближенное нахождение изолированных действительных корней выполняется в два этапа:

1. Нахождение приближенного значения корня – так называемого нулевого приближения.
2. Уточнение приближенного значения корня до тех пор, пока не будет достигнута заданная точность решения, путем итераций или последовательных приближений.

Остановимся подробно на втором этапе, так как нахождение нулевого приближения является специфической задачей, решаемой обычно либо на основе физических соображений или конструктивных особенностей изучаемого объекта.

Метод простых итераций (метод последовательных приближений).

В математике под итерацией понимается повторное применение какой-либо математической операции. В программировании итерация – организация обработки, при которой исходные данные изменяются при неизменно повторяющемся алгоритме.

Говорят, что итерационный процесс сходится, если при выполнении последовательных итераций получаются значения корней, все ближе и ближе приближающиеся к точному значению корня. В противном случае итерационный процесс считается расходящимся.

Перепишем для удобства уравнение $F(x) = 0$, в виде $x = f(x),$
что можно получить путем замены:

$$F(x) = x - f(x).$$

Пусть x_0 – нулевое приближение, т.е. начальное приближенное значение корня уравнения (3). Тогда в качестве следующего, 1-го, приближения примем

$$x_1 = f(x_0),$$

следующим, 2-м, приближением будет

$$x_2 = f(x_1),$$

и т.д., в качестве ***n***-го приближения примем

$$x_n = f(x_{n-1}).$$

Метод простых итераций основан на представлении уравнения в виде $x=f(x)$ и многократном применении итерационной формулы $x_{n+1}=f(x_n)$ до тех пор, пока соблюдается условие $|x_{n+1}-x_n| \geq \xi$. где ξ –заданная погрешность вычисления корня x.

Здесь возникает главный вопрос: приближается ли x_n к истинному решению уравнения при неограниченном возрастании n ? Иными словами, сходится ли итерационный процесс?

Уловия сходимости метода итераций: если при всех значениях x_n, вычисляемых в процессе решения задачи:

1. $|f'(x)|<1$, то итерационный процесс сходится;
2. $|f'(x)|>1$, то итерационный процесс расходится.

Если производная $f'(x)$ в некоторых точках x_i по модулю меньше 1, а в других точках x_j – больше 1, то ничего определенного о сходимости итерационного процесса сказать нельзя. Он может, как сходиться, так и расходиться.

Алгоритм решения нелинейных и трансцендентных уравнений методом простых итераций на языке Q-Basic.

В строке 70 записано выражение ***f(x)=SinX + 0,25*** (подпрограмма), соответствующая решению трансцендентного уравнения ***F(x)=X–SinX–0,25=0***

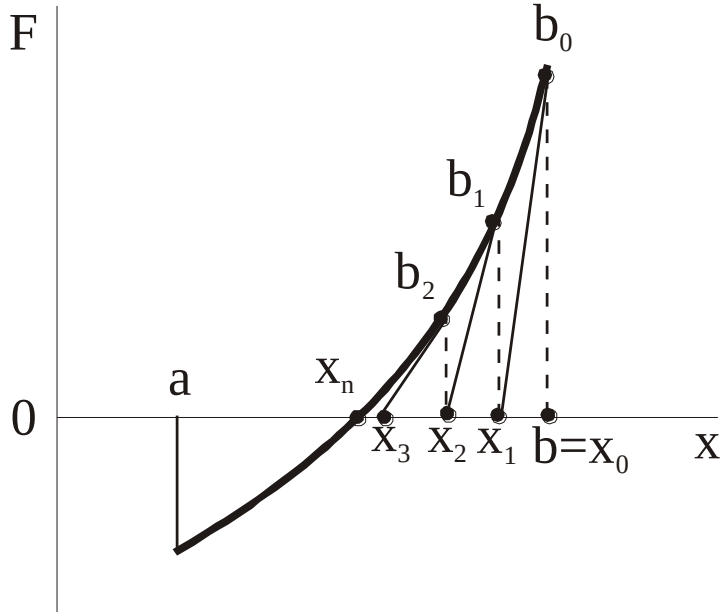
Для начального значения $X=X_0=1{,}2$ и погрешности $E=1*10^{-6}$ получим $X=1{,}171230492768914$

```
    Color 4,3
    Cls
    DefDbl A-Z
    Locate 8,15:? "Решение уравнения X=F(X) методом простых итераций"
    Locate 10,15:Color 0,3:Input "Задайте начальное значение X X0="X
    Locate 11,15:Input "Задайте погрешность результата E="E
30  GoSub 70
    If Abs(F-X)<E Then 60
```

```
        X=F
        GoTo 30
60      Cls
        Locate 13,15:Print "Корень уравнения X="X
        GoTo 20
        End
'Подпрограмма вычисления F(X)
70      F= Sin(X)+0.25
        Return
```

Метод Ньютона (касательных)

Метод Ньютона (касательных) основан на замене ***F(x)*** в точке начального приближения $x=x_0$ касательной, пересечение которой с осью x даёт первое приближение x_1, и т.д. Таким образом, итерационный процесс схождения к корню реализуется формулой $x_{n+1}=x_n-F(x_n)/F'(x_n)$ до тех пор, пока соблюдается условие $|x_{n+1}-x_n|\geq\xi$.. Метод обеспечивает быструю (квадратичную) сходимость, если $F(x_0)F''(x_0)>0$.



Рассмотрим вновь уравнение:

$$F(x)=0,$$

где функция $F(x)$ - дифференцируема и определена на некотором интервале $a<x<b$.

Разложим функцию $F(x)$ в степенной ряд и ограничимся линейной частью разложения:

$$F(x)=F(x_0)+F'(x_0)(x-x_0),$$

что эквивалентно замене функции ***F(x)*** в произвольной точке ***x*** ее касательной в этой точке.

Тогда следует:

$$x = x_0 - \frac{F(x_0)}{F'(x_0)}.$$

Если принять x_0 за нулевое приближение, то формулу можно использовать для нахождения следующего, 1-го приближения:

$$x_1 = x_0 - \frac{F(x_0)}{F'(x_0)},$$

отсюда следует, что **(*n*+1)**-е приближение определится по формуле:

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}.$$

Это и есть метод касательных или метод Ньютона-Рафсона.

Условия сходимости процесса (8) имеют вид:

1. нулевое приближение x_0 выбрано достаточно близко к корню уравнения $F(x) = 0$,
2. вторая производная ***F"(x)*** не становится слишком большой,
3. первая производная ***F'(x)*** не слишком близка к **0**.

Последнее условие означает, что никакие два корня не находятся близко друг от друга, а совместное выполнение условий 2) и 3) аналогично требованию $|f'(x)| < 1$ в методе простых итераций.

Процесс считается завершенным, если

$$\left|\frac{x_{n+1} - x_n}{x_n}\right| \le \varepsilon, \text{ где } \varepsilon$$

– заданная точность решения.

Метод Ньютона-Рафсона находит широкое применение для решения систем нелинейных уравнений высокого порядка.

Модифицированный метод Ньютона заключается в том, что вместо вычисления производной $F'(x_n)$ на каждом шаге итераций находится её приближённое значение $F'(x_n) = dF(x_n)/dx \approx (F(x_n+\Delta x)-F(x_n)) / \Delta x = \Delta F(x_n) / \Delta x$, где $\Delta x = \xi$. Следовательно итерационная формула имеет вид:

$$x_{n+1} = x_n - \frac{\Delta x F(x_n)}{F(x_n + \Delta x) - F(x_n)}.$$

Значение ***Δx*** не обязательно должно быть равно ξ. Равенство ***Δx=ξ*** позволяет уменьшить число исходных данных при вводе.

Алгоритм решения нелинейных и трансцендентных уравнений методом простых итераций на языке Q-Basic.

Для приведенного раннее примера X=1.171229652501929

```
        Color 4,3
        Cls
        DefDbl A-Z
        Locate 7,15:? "Решение уравнения F(X)=0 модифицированным"
        Locate 8,28:? "методом Ньютона"
20      Locate 10,15:Color 0,3:Input "Задайте начальное значение X0="X
        Locate 11,15:Input "Задайте погрешность результата E="E
30      GoSub 70
        L=F
        X=X+E
40      GoSub 70
        L=E*L/(F-L)
        X=X-L-E
50      If Abs(L)>E Then 30
60      Cls
        Locate 13,15:Print "Корень уравнения X="X
        GoTo 20
        End
'Подпрограмма вычисления F(X)
70      F=X-Sin(X)-0.25
        Return
```

Метод деления отрезка пополам (дихотомии)

Рассмотрим уравнение:

$F(x) = 0$,

где функция ***F(x)*** – непрерывна и определена на некотором отрезке $[a,b]$ и

$$F(a)F(b) < 0.$$

Последнее означает, что функция ***F(x)*** имеет на отрезке $[a,b]$ по крайней мере один корень. Рассмотрим случай, когда корень на отрезке $[a,b]$ единственный.

Делим отрезок пополам. Если $F\left(\frac{a+b}{2}\right) = 0$, то $\xi = \frac{a+b}{2}$ является корнем уравнения. Если $F\left(\frac{a+b}{2}\right) \neq 0$, то рассматриваем ту половину отрезка $[a,b]$, на

концах которой функция ***F(x)*** имеет разные знаки. Новый, более узкий отрезок $[a_1, b_1]$ вновь делим пополам и проводим на нем такое же рассмотрение и т.д. В результате на некотором шаге получим либо точное значение корня уравнения, либо последовательность вложенных друг в друга отрезков $[a_1, b_1], [a_2, b_2], \ldots, [a_n, b_n], \ldots$, таких, что

$$F(a_n)F(b_n) < 0, \quad (n = 1, 2, \ldots) \text{ и } b_n - a_n = \frac{b-a}{2^n}.$$

Левые концы этих отрезков $a_1, a_2, \ldots, a_n, \ldots$ образуют монотонную (неубывающую) ограниченную последовательность, а правые концы $b_1, b_2, \ldots, b_n, \ldots$ – монотонную (невозрастающую) ограниченную последовательность. Поэтому существует общий предел

$$\xi = \lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n.$$

При $n \to \infty$, в силу непрерывности функция ***F(x)*** получим: $[F(\xi)]^2 \le 0$. Отсюда $F(\xi) = 0$, т.е. ξ является корнем уравнения.

На практике процесс считается завершенным, если

$$b_n - a_n = \frac{b-a}{2^n} \le \varepsilon, \qquad (11)$$

где ε – заданная точность решения.

Алгоритм решения нелинейных и трансцендентных уравнений методом дихотомии на языке Q-Basic.

Для приведенного ранее примера ***X***=1.171229362487793 для ***A***=0, ***B***=2, ***H***=0.000001

```
      Color 4,3
      Cls
      DefDbl A-Z
      Locate 7,15:? "Решение уравнения F(X)=0 методом дихотомии"
10    Locate 9,15:Color 0,3:Input "Задайте начало интервала A="A
      Locate 10,15:Input "Задайте конец интервала B="B
      Locate 11,15:Input "Задайте погрешность результата H="H
      X=A
      GoSub 90
      F1=F
20    X=(A+B)/2
      GoSub 90
```

```
        If F1<0 Then
                If F<0 Then A=X Else B=X
        ElseIf F1>0 Then
                If F>0 Then A=X Else B=X
        End If
        If B-A>H Then 20
        Cls
        Locate 13,15:Print "Корень уравнения X="X
        GoTo 10
        End
'Подпрограмма вычисления F(X)
90      F=X-Sin(X)-0.25                 'F(a)<0 в интервале a=0, b=2
        Return
```

Метод хорд.

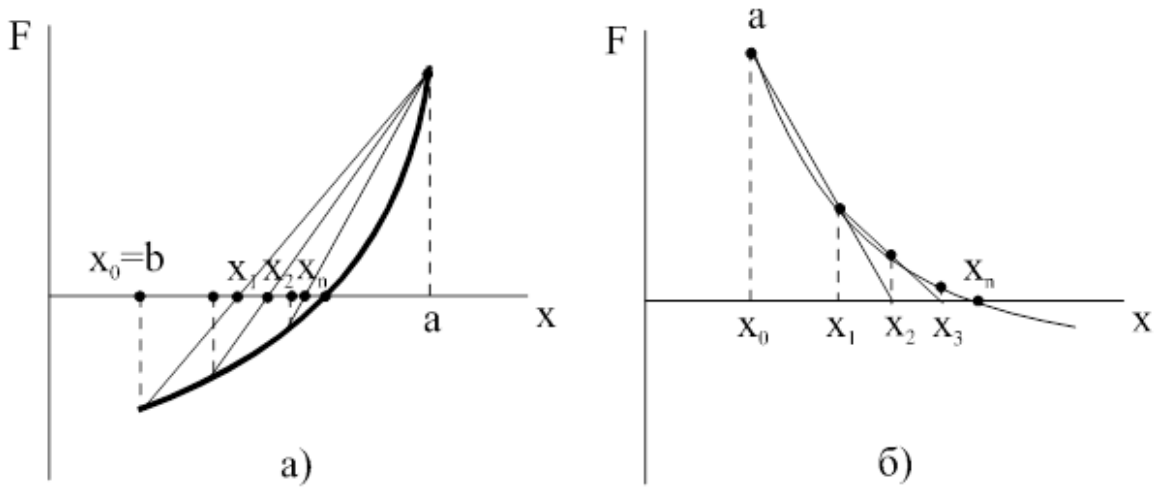
При этом методе каждое значение x_{n+1} находится как точка пересечения оси абсцисс с хордой, проведённой через точки ***F(a)*** и ***F(b)***, причём одна из этих точек, для которой знаки ***F(x)*** и ***F"(x)*** одинаковы – фиксируется. Если неподвижен конец хорды $x=a$, то

$$x_{n+1} = x_n - \frac{F(x_n)}{F(x_n) - F(a)}(x_n - a),$$

а если неподвижен конец хорды $x=b$, то

$$x_{n+1} = x_n - \frac{F(x_n)}{F(b) - F(x_n)}(b - x_n),$$

Если $|x_{n+1} - x_n| \geq \xi$, то в первом случае считаем $b=x_{n+1}$, во втором $a=x_{n+1}$ и повторяем вычисления. При использовании метода хорд полагается, что корень находится на отрезке ***[a,b]***.



Решение уравнения F(x)=0 методом хорд (а) и секущих (б).

Если $|x_{n+1}-x_n| < \xi$, итерации прекращаются, и x_{n+1} считается корнем уравнения.

Метод секущих.

Метод секущих реализуется алгоритмом, описанным выше, если абсциссы ***a*** и ***b*** взяты с одной стороны корня и не фиксируются.

Необходимость вычисления ***F'(x)*** (условия сходимости этих методов аналогичны) и выбора одной из двух формул затрудняют практическое применение методов хорд и секущих в отдельности.

Комбинированный метод секущих-хорд обеспечивает гарантированную сходимость при выборе в пределах отрезка ***[a,b]*** двух приближений: нулевого x_0 и первого x_1. Он реализуется алгоритмом метода Ньютона с заменой производной ***F'(x)*** её приближённым значением – множитель перед $F(x_n)$:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{F(x_n) - F(x_{n-1})} F(x_n).$$

Алгоритм решения нелинейных и трансцендентных уравнений методом дихотомии на языке Q-Basic.

Три примера функции заданы в подпрограмме со строки 80, в комментарии указаны контрольные ответы.

```
     Color 4,3
     Cls
     DefDbl A-Z
     Locate 6,15:? "Решение уравнения F(X)=0 комбинированным"
     Locate 7,25:? "методом секущих-хорд"
10   Locate 9,15:Color 0,3:Input "Задайте нулевое приближение X0="X0
     Locate 10,15:Input "Задайте первое приближение X1="X1
     Locate 11,15:Input "Задайте погрешность результата H="H
20   X=X0
     GoSub 80
     A=F
     X=X1
     GoSub 80
     B=F
     Y=X1-B*(X1-X0)/(B-A)
     X0=X1
     X1=Y
     If Abs(X1-X0)>H Then 20
     Cls
     Locate 13,15:Print "Корень уравнения X="X
     GoTo 10
     End
```

```
'Подпрограмма вычисления F(X)
80      'F=2-X-10^(-7)*(Exp(20*X)-1) 'При X0=0.7, H=0.0001 и X1=0.9 X=0.814398...
        'F=X^3+X^2-x-1          'При X0=2, X1=1.5,и H=0.00001 X=1.000000014330134
        F=X-Sin(X)-0.25 'При X0=0, X1=2,и H=0.00001 X=1.171229288230553
        Return
```

Контрольные понятия для изучения.

1. Понятие трансцендентного уравнения.
2. Итерационные методы.
3. Принципы окончания итераций.
4. Итерационные методы решения нелинейных и трансцендентных уравнений.

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В текстовом редакторе "Word" на основе приведенных алгоритмов создать процедуры нахождения корней нелинейных уравнений следующими методами:

- методом простых итераций;
- методом деления отрезка пополам;
- методом хорд;
- методом секущих;
- методом касательных.

Текст процедур занести в протокол.

ЛАБОРАТОРНАЯ РАБОТА №13

Численные методы. 6. Итерационные методы решения систем линейных уравнений.

Цель: практически ознакомиться с итерационными методами решения систем линейных уравнений.

Основные вопросы.

1. Метод простых итераций.
2. Метод Зейделя.
3. Решение системы линейных уравнений с переопределённой матрицей.

Система **m** линейных алгебраических уравнений с **n** неизвестными (или, линейная система, также употребляется аббревиатура СЛА́У) в линейной алгебре – это система уравнений вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Система называется **однородной**, если все её свободные члены равны нулю (**$b_1 = b_2 = \ldots = b_m = 0$**), иначе — **неоднородной.**

Система называется **совместной**, если она имеет хотя бы одно решение, и **несовместной**, если у неё нет ни одного решения.

Совместная система может иметь одно или более решений.

Система называется **квадратной**, если число **m** уравнений равно числу **n** неизвестных.

Совместная система называется **определённой**, если она имеет единственное решение; если же у неё есть хотя бы два различных решения, то она называется **недоопределённой**. Если уравнений больше, чем неизвестных, она называется **переопределённой.**

Системы линейных уравнений называются **эквивалентными**, если множество их решений совпадает, то есть любое решение одной системы одновременно является решением другой, и наоборот.

Системы решаются точными и итерационными методами. Точные методы дают точное решение за конечное число операций, если все они выполняются без погрешности. Число операций у итерационных методов зависит от заданной погрешности вычислений.

Метод простых итераций

При начальных приближениях x_{i0} $(i=1,2,...,N)$ вычисляем последовательные приближения по формуле простых итераций:

$$x_{i(j+1)} = x_{i(j)} - \frac{1}{a_{ij}}\left(\sum_{k=1}^{n} a_{ik}x_{k(j)} - b_i\right)$$

до тех пор, пока $x_{i(j+1)} - x_{i(j)} > \varepsilon$, где (j) – номер итерации, ε – заданная погрешность вычислений. Итерационный процесс сходится, если величина модуля каждого диагонального элемента матрицы A больше суммы модулей остальных элементов. Необходимо задать начальные приближения в виде вектора $[x_{i0}]$.

Алгоритм решения систем линейных уравнений методом простых итераций.

для системы ($N=3$)

$$\begin{cases} 4x_1 + 0{,}24x_2 - 0{,}08x_3 = 8, \\ 0{,}09x_1 + 3x_2 - 0{,}15x_3 = 9, \\ 0{,}04x_1 - 0{,}08x_2 + 4x_3 = 20. \end{cases}$$

задав $\varepsilon = E = 10^{-4}$ и начальные приближения $x_{10}=x_{20}=x_{30}=1$, получим по нижеприведенной программе: $x_1=1{,}9091$; $x_2=3{,}1949$; $x_3=5{,}0448$ и числе итераций $S=5$.

```
Cls
DefDbl A-Z
DefInt i,j,K,S,N
Locate 6,15:? "Решение системы из N линейных уравнений"
Locate 7,25:? "методом простых итераций"
Locate 9,15:Input "Задайте число уравнений N="N
Dim A(N,N),B(N),X(N),Z(N)
Input "Задайте погрешность вычисления E="E
For i=1 To N
        For j=1 To N
                Print "Введите A"i;","j
                Input A(i,j)
                If j=N Then
                        Print "Введите B"i
                        Input B(i)
```

```
                End If
            Next j
        Next i
        S=0
        For i=1 To N
            Print "Задайте X"i;"(0)"
            Input Z(i)
        Next i
100     K=0
        For i=1 To N
            X(i)=-B(i)
            For j=1 To N
                X(i)=X(i)+A(i,j)*Z(j)
            Next j
            If Abs(X(i)/A(i,i))>=E Then K=1
            X(i)=Z(i)-X(i)/A(i,i)
        Next i
        For i=1 To N
200         Z(i)=X(i)
        Next i
        S=S+1
        If K=1 Then 100
        Print "Результаты решения"
        For i=1 To N
            Print "X"i;"=";X(i)
        Next i
        Print "Число итераций S="S
        End
```

Метод Зейделя

При *итерационном методе Зейделя* процесс подобен описанному для метода простых итераций. Однако уточнённые значения $x_{i(j+1)}$ сразу подставляются в последующие уравнения. Формула итерационного процесса имеет вид:

$$x_{i(j+1)} = x_{i(j)} - \frac{1}{a_{ij}}\left(\sum_{k=1}^{i-1} a_{ik}x_{k(j+1)} + \sum_{k=i}^{N} a_{ik}x_{k(j)} - b_i\right)$$

Обычно метод Зейделя сходится быстрее, чем метод простых итераций, но возможна и обратная ситуация.

Алгоритм решения систем линейных уравнений методом Зейделя.

Изменив положение оператора 200 в предыдущей программе получим реализацию метода Зейделя. Для приведенного примета сходимость будет отличаться на 1.

```
100     K=0
```

```
        For i=1 To N
                X(i)=-B(i)
                For j=1 To N
                        X(i)=X(i)+A(i,j)*Z(j)
                Next j
                If Abs(X(i)/A(i,i))>=E Then K=1
                X(i)=Z(i)-X(i)/A(i,i)
200             Z(i)=X(i)
        Next i
        S=S+1
        If K=1 Then 100
        Print "Результаты решения"
```

Решение системы линейных уравнений с переопределённой матрицей.

Решается система ***AX=B***, где

$$A=\begin{bmatrix} a_{11}\ a_{21}\ \dots\ a_{1n} \\ a_{21}\ a_{22}\ \dots\ a_{nn} \\ \dots\dots\dots\dots\dots \\ a_{m1}\ a_{m1}\ \dots\ a_{mn} \end{bmatrix}, \qquad B=\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$$

и $\boldsymbol{m \geq n}$ (т.е. в общем случае число уравнений в системе больше, чем число неизвестных). Система может быть несовместимой. Её решением считается вектор неизвестных, при котором скаляр ***(AX–B) (AX–B)*** принимает наименьшее значение. Решение сводится к решению системы **[*A'A*][*X*]=[*AB*]**, где ***A'*** – транспонированная (замена строк на столбцы) матрица ***A***. Для этого используется метод квадратных корней.

Решение системы ***AX=B*** (в нашем случае $\boldsymbol{A \leftarrow A'A}$ и $\boldsymbol{B \leftarrow A'B}$) методом квадратных корней реализуется по следующим формулам прямого хода:

$$a_{11}^{(1)}=\sqrt{a_{11}}, \qquad a_{1j}^{(1)}=a_{1j}/a_{11}^{(1)}$$

$$f_1^{(1)}=f_1/a_{11}^{(1)}, \qquad j=1,2,3,\dots,n,$$

$$a_{ii}^{(1)}=\sqrt{a_{ii}-\sum_{k=1}^{i-1}a_{ki}^{(1)2}}, \qquad i=1,2,3,\dots,n,$$

$$a_{ij}^{(1)} = \left(a_{ij} - \sum_{k=1}^{i-1} a_{ki}^{(1)} a_{kj}^{(1)} \right) / a_{ij}^{(1)}, \quad j > i,$$

$$f_i^{(1)} = \left(f_i - \sum f_k^{(1)} a_{kj}^{(1)} \right) / a_{ii}^{(1)}, \quad x_n = f_n^{(1)} / a_{nn}^{(1)}.$$

Затем проводят обратный ход:

$$x_i = \left(f_i^{(1)} - \sum_{k=i+1}^{n} a_{ik}^{(1)} x_k \right) / a_{ii}^{(1)}, \quad i = n, n-1, \ldots, 1.$$

при ***m=n*** этим методом возможно решение обычных систем линейных уравнений.

Алгоритм решения систем линейных уравнений с переопределённой матрицей.

Для системы уравнений (n=N=3, m=M=6)

$$\begin{cases} 0{,}41x_1 - 0{,}35x_2 + 0{,}16x_3 = 0{,}22, \\ 0{,}23x_1 + 0{,}2x_2 + 0{,}41x_3 = 0{,}84, \\ 0{,}46x_1 - 0{,}13x_2 + 0{,}48x_3 = 0{,}81, \\ 0{,}17x_1 + 0{,}45x_2 - 0{,}25x_3 = 0{,}37, \\ 0{,}83x_1 + 0{,}27x_2 - 0{,}51x_3 = 0{,}59, \\ 0{,}27x_1 + 0{,}82x_2 - 0{,}14x_3 = 0{,}95 \end{cases}$$

получим $x_1=1$, $x_2=1$ и $x_3=1$

```
Cls
DefDbl A-Z
DefInt i,j,K,S,N
Locate 6,15:? "Решение переопределенной системы линейных уравнений"
Locate 7,25:? "методом квадратных корней"
'Ввод данных
Locate 9,15:Input "Введите число неизвестных N="N
Locate 10,15:Input "Введите число уравнений M="M
Dim C(N),A(M,N),B(M)
For i=1 To M
    For j=1 To N
        Print "Введите A";i;j
        Input A(i,j)
        If j=N Then
            Print "Введите B";i
            Input B(i)
        End If
    Next j
Next i
```

```
'Преобразование матрицы
80      For j=1 To N
            For k=j To N
                S=0
                For i=1 To M
                    S=S+A(i,j)*A(i,k)
                Next i
                C(k)=S
            Next k
            C=0
            For i=1 To M
                C=C+A(i,j)*B(i)
            Next i
            For i=j To N
                A(i,j)=C(i)
            Next i
            C(j)=C
        Next j
'Решение системы
        A(1,1)=Sqr(A(1,1))
        For j=2 To N
            A(1,j)=A(j,1)/A(1,1)
        Next j
        For i=2 To N
            S=0
            For k=1 To i-1
                S=S+A(k,i)*A(k,i)
            Next k
            A(i,i)=Sqr(A(i,i)-S)
            For j=i+1 To N
                S=0
                For k=1 To i-1
                    S=S+A(k,i)*A(k,j)
                Next k
                A(i,j)=(A(j,i)-S)/A(i,i)
            Next j
        Next i
        C(1)=C(1)/A(1,1)
        For i=2 To N
            S=0
            For k=1 To i-1
                S=S+A(k,i)*C(k)
            Next k
            C(i)=(C(i)-S)/A(i,i)
        Next i
210     C(N)=C(N)/A(N,N)
        For i=N-1 To 1 Step -1
            S=0
            For k=i+1 To N
```

```
            S=S+A(i,k)*C(k)
        Next k
        C(i)=(C(i)-S)/A(i,i)
Next i
Print "Решение системы"
For i=1 To N
        Print "X";i;"=";C(i)
Next i
End
```

Контрольные понятия для изучения.

5. Понятие систем линейных алгебраических уравнений.
6. Решение СЛАУ методом простых итераций.
7. Решение СЛАУ методом Зейделя.
8. Решение системы линейных уравнений с переопределённой матрицей.

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В текстовом редакторе "Word" на основе приведенных алгоритмов создать процедуры решения систем линейных алгебраических уравнений следующими методами:

- методом простых итераций;
- методом Зейделя;
- решение системы линейных уравнений с переопределённой матрицей.

Тексты созданных процедур занести в протокол.

ЛАБОРАТОРНАЯ РАБОТА №14

Численные методы. Численное дифференцирование.

Цель: практически ознакомиться с методами вычисления значения производной дискретно (таблично) заданной функции.

Основные вопросы.

1. Понятие численного дифференцирования.
2. Первая производная. Двухточечные методы.
3. Вычисление первых производных по трёхточечным схемам.
4. Дифференцирование основе первой интерполяционной формулы Ньютона.
5. Численное дифференцирование при равномерно расположенных узлах.
6. Формулы численного дифференцирования (производные в центральных узлах).

Производная функции есть предел отношения приращения функции к приращению независимой переменной при стремлении к нулю приращения независимой переменной

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

При численном нахождении производной заменим отношение бесконечно малых приращений функций и аргумента $\frac{dy}{dx}$ отношением конечных разностей. Очевидно, что чем меньше будет приращение аргумента, тем точнее численное значение производной.

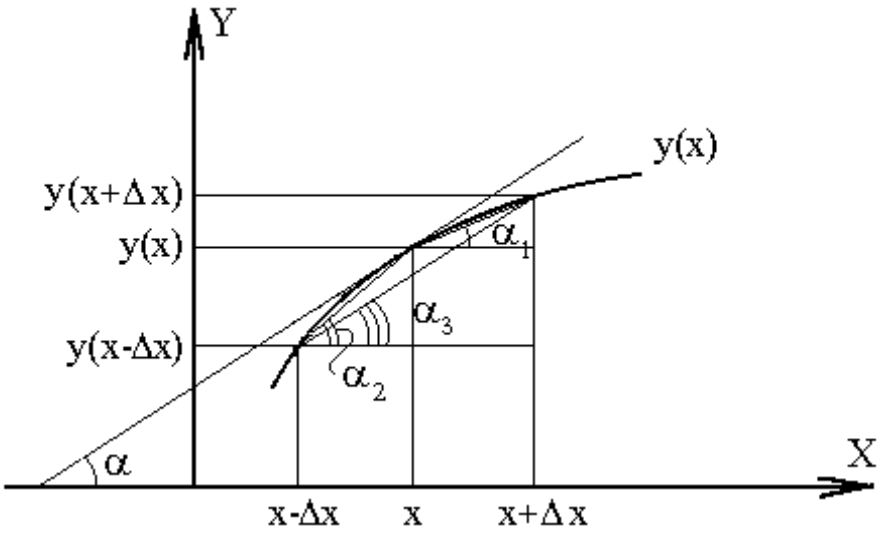
При решении инженерно-технических и других прикладных задач часто бывает необходимо найти производную определенного порядка от функции $f(x)$, заданной таблично. Кроме того, иногда в силу сложности аналитического выражения функции $f(x)$ ее непосредственное дифференцирование слишком затруднительно. В этих случаях обычно используют **численное дифференцирование**.

Первая производная. Двухточечные методы.

Для двухточечных методов при вычислении производных используется значение функции в двух точках. Приращение аргумента задается тремя способами, откладывая Δx=h вправо, влево и в обе стороны от исследуемой точки. Соответственно получается три двухточечных метода численного дифференцирования:

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y(x+\Delta x) - y(x)}{\Delta x};$$
$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y(x) - y(x-\Delta x)}{\Delta x};$$
$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y(x+\Delta x) - y(x-\Delta x)}{2\Delta x}.$$

Суть указанных методов проиллюстрирована на рисунке. Численное значение тангенса угла **α** образованного касательной к графику **y(x)** и осью абсцисс, показывает точное значение производной (геометрический смысл производной). Тангенсы углов $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, $\boldsymbol{\alpha_3}$ соответствуют приближенным значениям производных, определенных тремя методами соответственно.



Пример.

Вычислить точное и приближенное (тремя методами) значения производной функции $\mathbf{y=x^2}$ в точке **x=1** с шагом **h=1** и **h=0.001**.

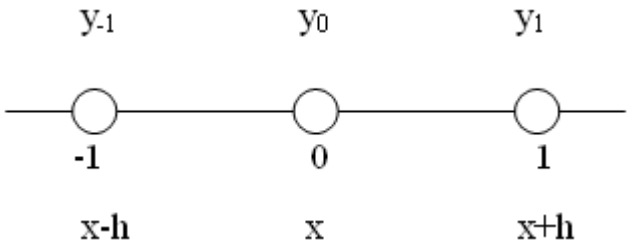
Для шага h=1:

$$y_1' = \frac{y(1+1) - y(1)}{1} = \frac{4-1}{1} = 3;$$
$$y_2' = \frac{y(1) - y(1-1)}{1} = \frac{1-0}{1} = 1;$$
$$y_3' = \frac{y(1+1) - y(1-1)}{1*2} = \frac{4-0}{2} = 2.$$

Для шага h=001:

$$y_1' = \frac{1.001*1.001 - 1*1}{0.001} = 2.001;$$
$$y_2' = \frac{1*1 - 0.999*0.999}{0.001} = 1.999;$$
$$y_3' = \frac{1.001*1.001 - 0.999*0.999}{0.001*2} = 2.$$

Вычисление первых производных по трёхточечным схемам.



Расчетные формулы для указанной на рисунке трехточечной схемы имеют вид:

$$y_{-1}' = \frac{(-3*y_{-1} + 4*y_0 - y_1)}{2*h};$$
$$y_0' = \frac{(-y_{-1} + 0*y_0 + y_1)}{2*h};$$
$$y_1' = \frac{(y_{-1} - 4*y_0 + 3*y_1)}{2*h}.$$

Дифференцирование основе первой интерполяционной формулы Ньютона.

Для нахождения первой и второй производных функции y=f(x) функцию y, заданную в равноотстоящих точках (i = 0, 1, 2, …, n) отрезка [a, b] значениями $y_i=f(x_i)$, приближенно заменяют интерполяционным многочленом Ньютона, построенным для системы узлов x_i:

$$y(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \frac{q(q-1)(q-2)}{3!}\Delta^3 y_0 + \frac{q(q-1)(q-2)(q-3)}{4!}\Delta^4 y_0 + ...,$$

где

$h = x_{i-1} - x, (i = 0,1,2,...),$

$q = \frac{x - x_0}{h}.$

Раскрывая скобки и учитывая, что

$$\frac{dy}{dx} = \frac{dy}{dq} * \frac{dq}{dx} = \frac{1}{h} * \frac{dy}{dq},$$

получим:

$$y'(x) = \frac{1}{h}\left[\Delta y_0 + \frac{2q-1}{2} * \Delta^2 y_0 + \frac{3q^2 - 6q + 2}{6} * \Delta^3 y_0 + \frac{2q^3 - 9q^2 + 11q - 3}{12} * \Delta^4 y_0 + ...\right]$$

Аналогично, учитывая

$$y'' = \frac{d(y')}{dx} = \frac{d(y')}{dq} * \frac{dq}{dx} = \frac{1}{h} * \frac{d(y')}{dq},$$

получим:

$$y''(x) = \frac{1}{h^2}\left[\Delta^2 y_0 + (q-1) * \Delta^3 y_0 + \frac{6q^2 = 18q + 11}{12} * \Delta^4 y_0 + ...\right]$$

Таким же образом можно при необходимости вычислить производную функции **y(x)** любого порядка. Заметим, что при вычислении производных в фиксированной точке **x** в качестве $\mathbf{x_0}$ следует брать ближайшее табличное значение аргумента.

Численное дифференцирование при равномерно расположенных узлах.

Численное дифференцирование при равномерно расположенных узлах с интерполяцией реализуется следующими формулами (для 3, 4 и 5 узлов):

$$y'(x_0 + ph)_3 = \frac{1}{h}\left[(p - 0{,}5)y_{-1} - 2py_0 + (p + 0{,}5)y_1\right],$$

$$y'(x_0 + ph)_4 = \frac{1}{h}\left(-\frac{3p^2 - 6p + 2}{6}y_{-1} + \frac{3p^2 - 4p - 1}{2}y_0 - \frac{3p^2 - 2p - 2}{2}y_1 + \frac{3p^2 - 1}{6}y_2\right),$$

$$y'(x_0 + ph)_4 = \frac{1}{h}\left(\frac{2p^3 - 3p^2 - p + 1}{12}y_{-2} - \frac{4p^3 - 3p^2 - 8p + 4}{6}y_{-1} + \right.$$

$$\left. + \frac{2p^3 - 5p}{2}y_0 - \frac{4p^3 + 3p^2 - 8p - 4}{6}y_1 + \frac{2p^3 + 3p^2 - p - 1}{12}y_2\right).$$

В этих формулах $p=(x-x_0)/h$ и $x=x_0+ph$.

Вычисление $y'(x)$ по последней формуле реализовано ниже:

```
        Color 4,3
        Cls
        DefDbl A-Z
        DefInt n
        Locate 4,25:? "Вычисление первой производной"
        Locate 5,14:? "таблично заданной функции при равномерном расположении"
        Locate 6,19:? "узлов интерполяции и числом пар данных от 3 до 5"
        Color 0,3:Locate 8,14:Input "Введите минимальное значение аргумента"X0
        Locate 9,14:Input "Введите шаг изменения аргумента"h
1       Locate 10,14:Input "Введите число пар данных"n
        On n-2 GoTo 3,4,5
        ? "Повторите ввод (число пар данных=3 или 4 или 5)"
        GoTo 1
3       Input "Введите значение функции Y1, Y2, Y3"Y1,Y2,Y3
        GoTo 2
4       Input "Введите значение функции Y1, Y2, Y3, Y4"Y1,Y2,Y3,Y4
        GoTo 2
5       Input "Введите значение функции Y1, Y2, Y3, Y4, Y5"Y1,Y2,Y3,Y4,Y5
2       Input "Введите значение аргумента X"X
        p=(X-X0)/h
        On n-3 GoTo 6,7
        f=((p-0.5)*Y1-2*p*Y2+(p+0.5)*Y3)/h
8       ? "dY/dX="f
        ? "Продолжить вычисления (Y/N)
9       If Inkey$="Y" Or Inkey$="y" Then 2
        If Inkey$="N" Or Inkey$="n" Then 10
        GoTo 9
6       f1=(3*p^2-4*p-1)*Y2/-(3*p^2-6*p+2)*Y1/6
        f2=(3*p^2-1)*Y4/6-(3*p^2-2*p-2)*Y3/2
        f=(f1+f2)/h
        GoTo 8
7       f1=(((2*p-3)*p-1)*p+1)*Y1/12
        f2=(((4*p-3)*p-8)*p+4)*Y2/6
        f3=(2*p^2-5)*p*Y3/2
        f4=(((4*p+3)*p-8)*p-4)*Y4/6
        f5=(((2*p+3)*p-1)*p-1)*Y5/12
        f=(f1-f2+f3-f4+f5)/h
        GoTo 8
10      Color 4,3
        Cls
        Locate 15,15:? "Конец"
        End
```

Частные производные функции $f(x_1,x_2,...,x_n)$ вычисляются по приведенным выше формулам, если задавать приращение одной из переменных и оставлять неизменными (равными заданным значениям) остальные переменные.

Анализ чувствительности функции $f(x_1, x_2, ..., x_n)$ к изменению её параметров $x_1, x_2, ..., x_n$ основан на вычислении абсолютного приращения функции по формуле:

$$\Delta f\left(x_1, x_2, \ldots, x_n\right) = \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial f}{\partial x_n}\Delta x_n$$

где коэффициенты $\partial f / \partial x_i$ являются абсолютными коэффициентами чувствительности к изменению параметра x_i. Применяются и относительные коэффициенты чувствительности:

$$S_i = \frac{\partial f / f}{\partial x_i / x_i} = \lim_{\Delta x_i \to 0} \frac{\Delta f / f}{\Delta x_i / x_i}.$$

С их помощью легко вычисляется относительное приращение функции при заданных относительных изменениях параметров (переменных) $\Delta x_i / x_{i0}$:

$$\frac{\Delta f}{f\left(x_{10}, x_{20}, \ldots, x_{n0}\right)} = S_1 \frac{\Delta x_1}{x_{10}} + S_2 \frac{\Delta x_2}{x_{20}} + \ldots + S_n \frac{\Delta x_n}{x_{n0}}$$

Формулы численного дифференцирования (производные в центральных узлах).

Число узлов	Производные
3	$y'=(y_1-y_{-1})/2h$ $y''(y_1-2y_0+y_{-1})/h^2$
5	$y'=(-y_2+8y_1-8y_{-1}+y_{-2})/12h$ $y''=(-y_2+16y_1-30y_0+16y_{-1}-y_{-2})/12h^2$ $y'''=(y_2-2y_1+2y_{-1}-y_{-2})/2h^3$
7	$y'=(y_3-9y_2+45y_1-45y_{-1}+9y_{-2}-y_{-3})/60h$ $y''=(2y_3-27y_2+270y_1-490y_0+270y_{-1}-27y_{-2}+2y_{-3})/180h^2$ $y'''=(-y_3+8y_2-13y_1+13y_{-1}-8y_{-2}+y_{-3})/8h^3$

Вычисление трёх частных производных и относительных коэффициентов чувствительности функции нескольких переменных можно организовать следующим образом (Программа 3.2):

1. Будем считать переменные $x_1, x_2, ..., x_n$ переменными массива ***X(I)*** и организуем ввод числа переменных ***N*** и начальных значений переменных $x_{10}, x_{20}, ..., x_{n0}$.
2. Вводим приращение $\Delta x = H$.
3. Организуем цикл вычисления частных производных по каждой переменной с управляющей переменной ***I=1,2,...,N***. Внутри цикла задаём $x_i=(x_{i0}-2h), (x_{i0}-h),$

..., (x$_{i0}$+2h) (для 5 узлов) и для этих значений x_i, обращаясь к подпрограмме вычисления $f(x_1,x_2,...,x_n)$, находим ординаты $f_{-2}=y_{-2}$, $f_{-1}=y_{-1}$, ..., $f_2=y_2$. После этого вычисляем производные $f'(x_{i0})=y'(x_{i0})$, $f''(x_{i0})=y''(x_{i0})$, $f'''(x_{i0})=y'''(x_{i0})$ по формулам таблицы 3.1 и коэффициенты S_i и выводим их на печать.

4. После выхода из цикла идём к пункту 2 и повторяем вычисления (если это необходимо).

При N=1 по этому алгоритму вычисляется три производные и коэффициент S функции одной переменной $f(x)$.

```
    Color 4,3
   Cls
    Locate 4,22
10  Print "Вычисление трех частных производных"
    Locate 5,17
20  Print "и относительного коэффициента чувствительности"
    Locate 6,15
30  Print "функции F(Xi) ряда переменных, заданной 5 ординатами"
    Locate 8,17
    Print "(Вид функции вводится в программу со строки 400)"
    DefDbl A-Z
    DefInt i,N
40  Color 0,3:Locate 15,20
50  Input "Введите число переменных N="N
60  Dim A(N),B(N),C(N),S(N),X(N)
    For i=1 To N
        Locate 15,20
70      Print "Введите начальное значение X("i")";
80      Input X(i)
    Next i
90  Cls
    Locate 5,20
    Input "Задайте абсолютное приращение Xi H="H
100 For i=1 To N
        X(0)=X(i)
110     X(i)=X(0)-2*H
        GoSub 400
            F2=F
            GoSub 400
                F1=F
                GoSub 400
120                 F0=F
                    GoSub 400
                        E1=F
                        GoSub 400
        E2=F
130     X(i)=X(0)
```

```
            A(i)=(-E2+8*E1+-8*F1+F2)/12/H
140         S(i)=A(i)*X(i)/F0
150         B(i)=(-E2+16*E1-30*F0+16*F1-F2)/12/H/H
160         C(i)=(E2-2*E1+2*F1-F2)/2/H/H/H
    Next i
200 For i=1 To N
            Print "Для переменной X("i")","d1F/dX1="A(i)
220         Print "              d2F/dX2="B(i)
            Print "              d3F/dX3="C(i)
230         Print "                 S="S(i)
    Next i
240 Print "Значение функции F(X10)="F0
250 Print "Нужно расчитать при другом абсолютном приращении (Y/N)"
255 If Inkey$="Y" Or Inkey$="y" Then 90
    If Inkey$="N" Or Inkey$="n" Then 260
    GoTo 255
260 Color 4,3
    Cls
    Locate 11,17
    Print "Конец, нажмите любую клавишу."
    End
400         'ПОДПРОГРАММА ОПИСАНИЯ ДИФФЕРЕНЦИРУЕМОЙ ФУНКЦИИ
                F=EXP(-X(1)^2/2)/SQR(2*3.14)
410             X(i)=X(i)+H
                Return
```

Контрольные понятия для изучения.

1. Понятие численного дифференцирования.
2. Первая производная. Двухточечные методы.
3. Вычисление первых производных по трёхточечным схемам.
4. Дифференцирование основе первой интерполяционной формулы Ньютона.
5. Численное дифференцирование при равномерно расположенных узлах.
6. Формулы численного дифференцирования (производные в центральных узлах).

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В текстовом редакторе "Word" на основе приведенных алгоритмов создать процедуры нахождения производных по приведенным примерам программ, адаптировав их к среде Visual Basic.

Тексты процедур занести в протокол.

ЛАБОРАТОРНАЯ РАБОТА №15

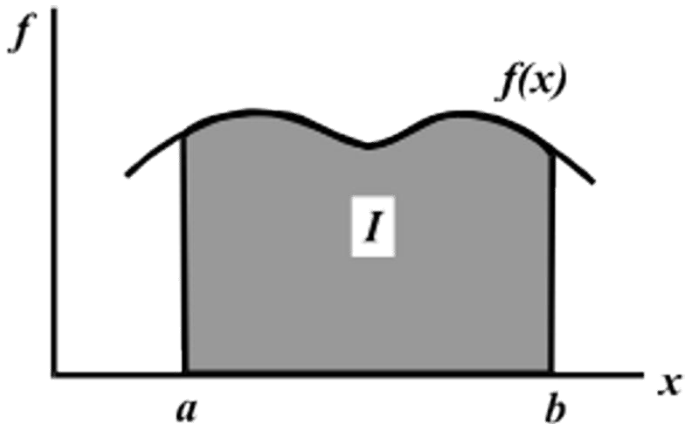
Численные методы. Численное интегрирование.

Цель: практически ознакомиться с численными методами вычисления определённых интегралов.

Основные вопросы.

1. Понятие численного интегрирования.
2. Метод прямоугольников.
3. Модифицированный метод прямоугольников.
4. Метод трапеций..
5. Метод Симпсона (парабол).
6. Методы Ньютона-Котеса.
7. Метод Гауса
8. Метод Монте-Карло

Определённый интеграл с пределами интегрирования **a** и **b** можно трактовать как площадь фигуры, ограниченной ординатами **a** и **b**, с осью абсцисс x и графиком подинтегральной функции **f(x)**.



Обыкновенный определённый интеграл, у которого известна его первообразная **F(x)**, вычисляется по формуле Ньютона-Лейбница

I = F(b) – F(a)

Поэтому, достаточно вычислить значения функции **F(x)**.

Идея численного интегрирования предельно проста и вытекает из геометрического смысла определенного интеграла – значение определенного интеграла численно равно площади криволинейной трапеции, ограниченной графиком функции **y=f(x)**, осью абсцисс и прямыми **x=a, x=b**. Находя приближенно

площадь криволинейной трапеции, мы получаем значение интеграла. Формально процедура численного интегрирования заключается в том, что отрезок **[a, b]** разбивается на **n** частичных отрезков, а затем подинтегральная функция заменяется на нем легко интегрируемой функцией, по определенной зависимости интерполирующей значения подинтегральной функции в точках разбиения.

Численное интегрирование применяется, если нахождение **F(x)** сложно или невозможно. Оно заключается в интерполяции **F(x)** на отрезке **[a, b]** подходящим полиномом, для которого определённый интеграл вычисляется по формулам численного интегрирования. Обычно отрезок **[a, b]** разбивается на **m** частей, к каждой из которых применяется соответствующая простая формула. Таким образом, получают составные (сложные) формулы численного интегрирования.

Метод прямоугольников.

Функция y=f(x) интегрируема на сегменте [a,b] и требуется вычислить ее интеграл $\int_a^b f(x)dx$. Составим интегральную сумму для **f(x)** на сегменте **[a,b]**. Для этого разобьем сегмент [a,b] на n равных между собой частей с помощью точек: $x_1, x_2, \dots, x_k, \dots, x_{n-1}$.

Если длину каждой части мы обозначим через Δx, так что $\Delta x = \frac{b-a}{n}$, то для каждой точки x_k будем иметь: $x_k = a + k\Delta x$ ***(k=0, 1, 2, …, n).***

Обозначим теперь через y_k значение подинтегральной функции ***f(x)*** при $x = x_k = a + k\Delta x$ то есть положим $y_k = f(a + k\Delta x)$ ***(k=0, 1, …, n).***

Тогда суммы $\sum_{k=1}^{n} y_{k-1}\Delta x$ и $\sum_{k=1}^{n} y_k \Delta x$ будут интегральными для функции ***f(x)*** на отрезке **[a,b]**. (При составлении первой суммы мы рассматриваем значения функции ***y=f(x)*** в точках, являющихся левыми концами частичных сегментов, а при составлении второй суммы – в точках, являющихся правыми концами этих сегментов.)

По определению интеграла имеем:

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=1}^{n} y_{k-1}\Delta x \quad \text{и} \quad \int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{k=1}^{n} y_k \Delta x$$

Поэтому в качестве приближенного значения $\int_a^b f(x)dx$ естественно взять интегральную сумму $\sum_{k=1}^{n} y_{k-1}\Delta x$ и $\sum_{k=1}^{n} y_k \Delta x$,т.е. положить:

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} y_{k-1}\Delta x$$

а также

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} y_k \Delta x$$

т.е
$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_0 + y_1 + y_2 + \ldots + y_{n-1})$$

и
$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_1 + y_2 + y_3 + \ldots + y_n)$$

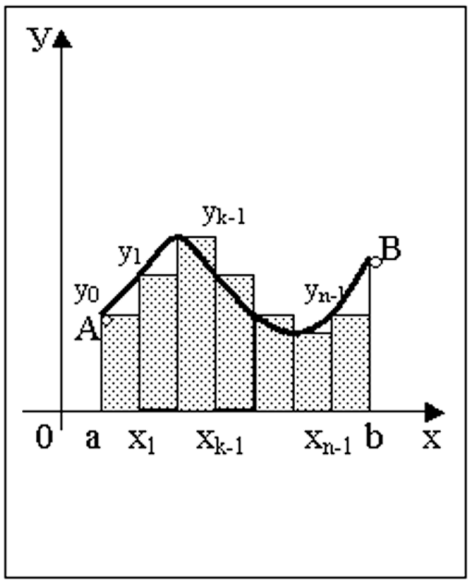
Эти приближенные равенства называются формулами прямоугольников.

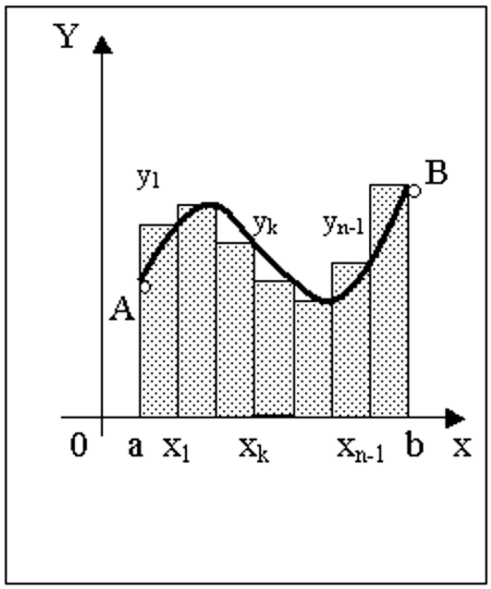
В том случае, когда $f(x) \geq 0$, формулы с геометрической точки зрения означают, что площадь криволинейной трапеции ***aABb***, ограниченной дугой кривой ***y=f(x)***, осью ***Ox*** и прямыми ***x=a*** и ***x=b***, принимается приближенно равной площади ступенчатой фигуры, образованной из **n** прямоугольников с основаниями $\Delta x = \frac{b-a}{n}$ и высотами: $y_0, y_1, y_2, \ldots, y_{n-1}$ и $y_1, y_2, y_3, \ldots, y_n$.

Исходя из приведенного выше геометрического смысла формул, способ приближенного вычисления определенного интеграла по этим формулам принято называть методом прямоугольников.

Всякое приближенное вычисление имеет определенную ценность лишь тогда, когда оно сопровождается оценкой допущенной при этом погрешности. Поэтому формулы прямоугольников будут практически пригодны для приближенного

вычисления интегралов лишь в том случае, если будет существовать удобный способ оценки получающейся при этом погрешности (при заданном **n**), позволяющий к тому же находить и число частей n разбиения сегмента, гарантирующее требуемую степень точности приближенного вычисления.





Будем предполагать, что функция ***f(x)*** имеет ограниченную производную на сегменте **[a, b]**, так что существует такое число ***M>0***, что для всех значений **x** из **[a, b]** выполняется неравенство $|f'(x)| \le M$. Качественный смысл этого неравенства заключается в том, что скорость изменения значения функции ограничена. В реальных природных системах это требование практически всегда выполнено. В этих условиях абсолютная величина погрешности $\mathbf{R_n}$, которую мы допускаем, вычисляя интеграл $\int_a^b f(x)dx$ по формуле прямоугольников может быть оценена по формуле:

$$|R_n| \le M(b-a)^2/2n$$

При неограниченном возрастании **n** выражение $M(b-a)^2/2n$, а следовательно, и абсолютная величина погрешности R_n будет стремиться к нулю, т.е. точность приближения будет тем больше, чем на большее число равных частей будет разделен сегмент **[a, b]**. Абсолютная погрешность результата будет заведомо меньше заданного числа $\varepsilon > 0$, если взять $n > M(b-a)^2/2\varepsilon$.

Следовательно, для вычисления интеграла $\int_a^b f(x)dx$ с указанной степенью точности достаточно сегмент **[a, b]** разбить на число частей, большее числа $M(b-a)^2/2\varepsilon$.

Метод прямоугольников – это наиболее простой и вместе с тем наиболее грубый метод приближенного интегрирования.

Модифицированный метод прямоугольников.

Модифицированный метод прямоугольников базируется на представлении ***f(x)*** ординатами, смещёнными на величину ***0,5h***, где ***h=(b–a)/m***:

$$I \cong h\sum f\left(x_i + 0{,}5h\right) + \frac{h^2(b-a)}{24} f''(\xi),$$

где ***f''(ξ)*** – значение второй производной ***f(x)*** в точке ***x=ξ***, где она максимальна.

Метод трапеций.

Если на частичном отрезке $[x_{j-1}, x_j]$ подынтегральную функцию заменить полиномом Лагранжа первой степени, то есть:

$$f(x) = L_{1,j}(x) = \frac{1}{h}\left[(x - x_{j-1})f(x_j) - (x - x_j)f(x_{j-1})\right]$$

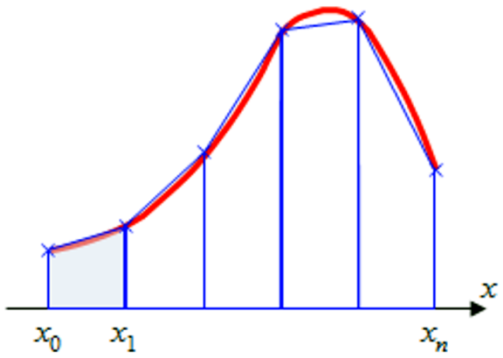
то искомый интеграл на частичном отрезке запишется следующим образом:

$$\int_{x_{j-1}}^{x_j} f(x)dx \approx \frac{1}{h}\left[f(x_j)\int_{x_{j-1}}^{x_j}(x - x_{j-1})dx - f(x_{j-1})\int_{x_{j-1}}^{x_j}(x - x_j)dx \right] = \frac{f(x_{j-1}) + f(x_j)}{2}h$$

И, составная формула трапеций на всем отрезке интегрирования $[a, b]$ примет вид:

$$\int_a^b f(x)dx \approx \sum_{j=1}^{N} \frac{f(x_j) + f(x_{j-1})}{2}h = h\left[\frac{1}{2}(f_1 + f_N) + f_2 + \ldots + f_{N-1}\right]$$

Графически площадь криволинейной трапеции заменяется площадью многоугольника, составленного из ***N*** трапеций, при этом кривая заменяется вписанной в нее ломаной. На каждом из частичных отрезков функция аппроксимируется прямой, проходящей через конечные значения.



Погрешность метода трапеций выше, чем у метода прямоугольников. Однако на практике использовать метод средних прямоугольников удается далеко не всегда.

Метод Симпсона (парабол).

В этом методе подынтегральная функция на частичном отрезке $[x_{j-1}, x_j]$ аппроксимируется параболой, проходящей через три точки x_{j-1}, $x_{j-0.5}$, x_j, то есть интерполяционным многочленом Лагранжа второй степени:

$$f(x) = L_{2,j}(x) = \frac{2}{h^2}\left[(x - x_{j-0.5})(x - x_j) f(x_{j-1}) - \right.$$
$$\left. - 2\cdot(x - x_{j-1})(x - x_j) f(x_{j-0.5}) + (x - x_{j-1})(x - x_{j-0.5}) f(x_j)\right]$$

Проведя интегрирование, получим:

$$\int_{x_{j-1}}^{x_j} f(x)dx \approx \frac{h}{6}\left(f_{j-1} + 4f_{j-0.5} + f_j\right)$$

Это и есть формула Симпсона или формула парабол. На отрезке $[a, b]$ формула Симпсона примет вид:

$$\int_a^b f(x)dx \approx \frac{h}{6}\left[f_0 + f_N + 2(f_1 + f_2 + \ldots + f_{N-1}) + 4(f_{0.5} + f_{1.5} + f_{2.5} + \ldots + f_{N-0.5})\right] =$$
$$= \frac{h}{6}\left[f_0 + f_N + 2\cdot\sum_{j=1}^{N-1} f_j + 4\cdot\sum_{j=0.5}^{N-0.5} f_j\right]$$

(2.14)

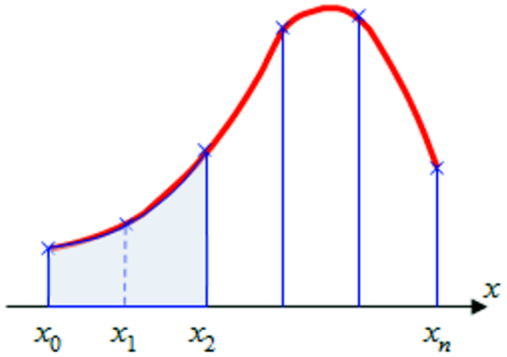
Можно разбить отрезок интегрирования $[a, b]$ на четное количество $2N$ равных частей с шагом $h = \frac{b-a}{2N}$. Тогда можно построить параболу на каждом сдвоенном

частичном отрезке $[x_{j-1}, x_j]$ и переписать выражения (2.12-2.14) без дробных индексов. Тогда формула Симпсона примет вид:

$$\int_a^b f(x)dx \approx \frac{h}{3}[f_0 + f_{2N} + 2(f_2 + f_4 + \ldots + f_{2N-2}) + 4(f_1 + f_3 + f_5 + \ldots + f_{2N-1})] =$$

$$= \frac{h}{3}\left[f_0 + f_{2N} + 2 \cdot \sum_{j=2,2}^{2N-2} f_j + 4 \cdot \sum_{j=1,2}^{2N-1} f_j \right]$$

Графическое представление метода Симпсона: на каждом из сдвоенных частичных отрезков заменяем дугу данной кривой параболой.



Пример:

Вычислить интеграл при **a=0** и **b=1**. Вычисление функции оформляется подпрограммой в строке **110**. Для **M=8** получим **I=1,39871724019002** , для **M=46** получим **I=1,398717474017935**

```
      Color 4,3
      Cls
      DefDbl A-Z
      DefInt M,N
10    Locate 8,15:? "Численное интегрирование методом Симпсона"
20    Locate 10,15:Color 0,3:Input "Введите нижний предел интегрирования A="A
25    Locate 11,15:Input "Введите верхний предел интегрирования B="B
30    Locate 13,15:Input "Введите число интервалов интегрирования M="M
40    H=(B-A)/M/2
      X=A
50    GoSub 110
      I=F
      N=0
60    X=X+H
      GoSub 110
      I=I+4*F
70    N=N+2
```

```
        If N=2*M Then 90
80      X=X+H
        GoSub 110
        I=I+2*F
        GoTo 60
90      X=B
        GoSub 110
        I=(I+F)*H/3
95      Cls
        Locate 14,15:Print "Для A="A;"; B="B;"; M="M
100     Locate 15,15:? "Значение интеграла I="I
        GoTo 30
        End
        'Подпрограмма вычисления подинтегральной функции
110     F=Sqr(2*X+1)
        Return
```

Методы Ньютона-Котеса.

Выше мы рассмотрели три схожих метода интегрирования функций – метод прямоугольников, метод трапеций, метод Симпсона. Их объединяет общая идея – интегрируемая функция интерполируется на отрезке интегрирования по равноотстоящим узлам многочленом Лагранжа, для которого аналитически вычисляется значение интеграла. Семейство методов, основанных на таком подходе, называется методами Ньютона-Котеса.

В выражении $\int_a^b f(x)\cdot dx \approx \sum_{j=1}^{N} c_j \cdot f(x_j)$ мы называли c_j коэффициентами, исходя из их смысла. Однако, правильнее эти величины называть весовыми коэффициентами. Величину $\Psi_N = \int_a^b f(x)dx - \sum_{j=0}^{N} c_j f(x_j)$, определяющую погрешность численного интегрирования, также называют остатком.

Для семейства методов Ньютона-Котеса можно записать общее выражение:

$$\int_a^b f(x)dx \approx \frac{n\cdot h}{C_n}\sum_{j=1}^{N}\sum_{i=0}^{n} c_{in} f(x_i)$$

где ***n*** – порядок метода Ньютона-Котеса, ***N*** – количество частичных отрезков, $h=\frac{x_j - x_{j-1}}{n}$, $C_n = \sum_{i=0}^{n} c_{in}$, $x_i = x_j + i\cdot h$.

Коэффициенты c_{in} могут быть заданы в табличной форме:

n	C_n	c_{0n}	c_{1n}	c_{2n}	c_{3n}	c_{4n}	c_{5n}
0	**1**	1					
1	**2**	1	1				
2	**6**	1	4	1			
3	**8**	1	3	3	1		
4	**90**	7	32	12	32	7	
5	**288**	19	75	50	50	75	19

Легко можно получить формулу прямоугольников для $n=0$, формулу трапеций для $n=1$, и формулу Симпсона для $n=2$.

Метод Гауса.

Метод Гауса основан на интерполяции ***f(x)*** полиномом Лагранжа, но абсциссы выбираются из условия обеспечения минимума погрешности интерполяции. При этом интеграл $I=\int_a^b f(x)dx$ подстановкой $x=\frac{b+a}{2}+\frac{b-a}{2}t$ сводится к виду:

$$I=\int f(t)dt=\sum A_i f(t_i).$$

Метод Гауса обычно обеспечивает повышенную точность и формула верна для полиномов до ***(2n-1)***-й степени. Для ***n=3***, ***A1=5/9***, ***t1***$=-\sqrt{1/3}$, ***A2=8/9***, ***t2=0***, ***A3=5/9*** и ***t3***$=\sqrt{1/3}$. Остаточный член (погрешность) при этом равен:

$$R_3=\frac{1}{15750}\left(\frac{b-a}{2}\right)^7 f^{VI}(\xi)$$

Для повышения точности интегрирования отрезок ***[a,b]*** дробится на ***m*** частей.

Пример:

Для задачи предыдущего примера, для **M=8** получим **I=1,398717474423129**, для **M=46** получим **I=1,398717474235549**

```
Color 4,3
Cls
DefDbl A-Z
DefLng E
Def FnSum(I)=I+Y
Def FnPr(A,B)=A*5*B/9
05  Locate 8,15:? "Численное интегрирование методом Гауса"
10  Locate 10,15:Color 0,3:Input "Введите нижний предел интегрирования A="Q
```

```
    12    Locate 11,15:Input "Введите верхний предел интегрирования B="B
    15    Locate 13,15:Input "Введите число интервалов интегрирования
M="M
          A=Q
    20    T=Sqr(0.6)
          I=0
          H=(B-A)/M
    30    For E=1 To M
                B1=A+H
                C=(B1+A)/2
                D=(B1-A)/2
    50          X=C-D*T
                      GoSub 130
    60          Y=FnPr(D,F)
                I=FnSum(I)
    70          X=C
                      GoSub 130
    80          Y=D*8*F/9
                I=FnSum(I)
    90          X=C+D*T
                      GoSub 130
    100         Y=FnPr(D,F)
                I=FnSum(I)
    110         A=B1
          Next E

    115   Cls
          Locate 14,15:Print "Для A="Q;"; B="B;"; M="M
    120   Locate 15,15:? "Значение интеграла I="I
          GoTo 15
          End
' Подпрограмма вычисления подинтегральной функции
    130   F=Sqr(2*X+1)
          Return
```

Метод Монте-Карло.

Сначала Энрико Ферми в 1930-х годах в Италии, а затем Джон фон Нейман и Станислав Улам в 1940-х в Лос-Аламосе предположили, что можно использовать связь между стохастическими (случайными) процессами и дифференциальными уравнениями «в обратную сторону». Они предложили использовать стохастический подход для аппроксимации многомерных интегралов в уравнениях переноса, возникших в связи с задачей о движении нейтрона в изотропной среде.

Идея была развита Уламом, который, раскладывая пасьянсы во время выздоровления после болезни, задался вопросом, какова вероятность того, что пасьянс сложится. Вместо того, чтобы использовать обычные для подобных задач

соображения комбинаторики, Улам предположил, что можно просто поставить эксперимент большое число раз и, подсчитав число удачных исходов, оценить вероятность. Он же предложил использовать компьютеры для расчётов методом Монте-Карло.

Годом рождения метода Монте-Карло считается 1949 год, Название метода происходит от названия коммуны в княжестве Монако, широко известного своими многочисленными казино, поскольку именно рулетка является одним из самых широко известных генераторов случайных чисел. Станислав Улам пишет в своей автобиографии «Приключения математика», что название было предложено Николасом Метрополисом в честь его дяди, который был азартным игроком.

Предположим, требуется вычислить определённый интеграл $\int_a^b f(x)\,dx$

Рассмотрим случайную величину u, равномерно распределённую на отрезке интегрирования $[a,b]$. Тогда $f(u)$также будет случайной величиной, причём её математическое ожидание выражается как $\mathbb{E}f(u) = \int_a^b f(x)\varphi(x)\,dx$, где $\varphi(x)$ — плотность распределения случайной величины u, равная $\frac{1}{b-a}$на участке $[a,b]$.

Таким образом, искомый интеграл выражается как

$$\int_a^b f(x)\,dx = (b-a)\mathbb{E}f(u)$$

.

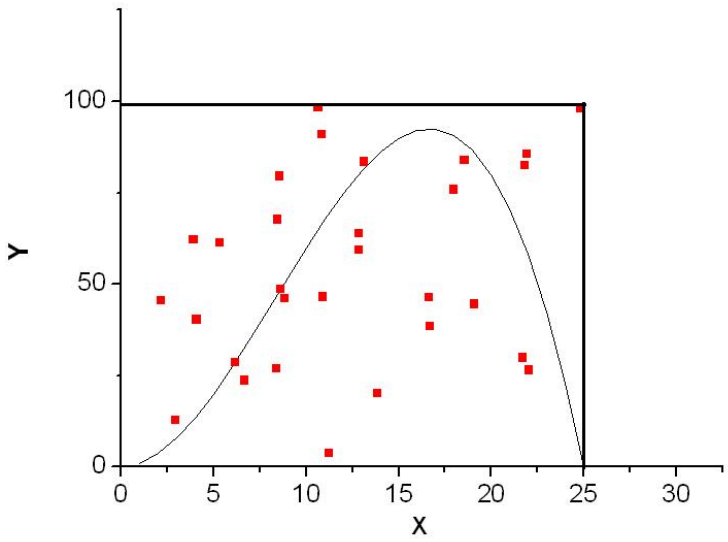
Но математическое ожидание случайной величины $f(u)$ можно легко оценить, смоделировав эту случайную величину и посчитав выборочное среднее.

Итак, бросаем Nточек, равномерно распределённых на $[a,b]$, для каждой точки u_i вычисляем $f(u_i)$. Затем вычисляем выборочное среднее: $\frac{1}{N}\sum_{i=1}^{N} f(u_i)$.

В итоге получаем оценку интеграла: $\int_a^b f(x)\,dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i)$

Точность оценки зависит только от количества точек N.

Геометрически алгоритм Монте-Карло выглядит следующим образом:



- ограничим функцию прямоугольником (*n*-мерным параллелепипедом в случае многих измерений), площадь которого S_{par}можно легко вычислить; любая сторона прямоугольника содержит хотя бы 1 точку графика функции, но не пересекает его;
- «набросаем» в этот прямоугольник (параллелепипед) некоторое количество точек (N штук), координаты которых будем выбирать случайным образом;
- определим число точек (K штук), которые попадут под график функции;
- площадь области, ограниченной функцией и осями координат, S даётся выражением $S = S_{par}\frac{K}{N}$

Для малого числа измерений интегрируемой функции производительность Монте-Карло интегрирования гораздо ниже, чем производительность детерминированных (конкретных, однозначных) методов. Тем не менее, в некоторых случаях, когда функция задана неявно, а необходимо определить область, заданную в виде сложных неравенств, стохастический (случайных значений) метод может оказаться более предпочтительным.

Контрольные понятия для изучения.

1. Понятие численного интегрирования.
2. Метод прямоугольников.
3. Модифицированный метод прямоугольников.
4. Метод трапеций..
5. Метод Симпсона (парабол).

6. Методы Ньютона-Котеса.
7. Метод Гауса
8. Метод Монте-Карло

Порядок выполнения.

Изучить теоретическую часть и занести в протокол основные положения.

В текстовом редакторе "Word" на основе приведенных алгоритмов создать процедуры нахождения производных по приведенным примерам программ, адаптировав их к среде Visual Basic.

Тексты процедур занести в протокол.

ТЕХНИКА БЕЗОПАСНОСТИ

Общие положения

1. К работе в учебной аудитории допускаются лица, ознакомившиеся с Инструкцией по технике безопасности и правилам поведения студентов в учебных аудиториях.
2. Выполнение инструкции обязательно для всех студентов.
3. Нельзя заходить и находиться в учебной аудитории без разрешения преподавателя.
4. Работа в учебной аудитории должна проводиться только в строгом соответствии с расписанием.

Требования безопасности перед началом работы

1. Студенты должны приходить за 5-10 минут до начала занятия, чтобы заблаговременно подготовиться к нему.
2. Начинать работу за компьютером можно только по приглашению преподавателя.
3. Студенты должны быть осторожными при подключении аппаратуры в электросеть.

Требования безопасности во время работы

1. Студенты должны бережно относиться к компьютерной технике и другому оборудованию учебной аудитории.
2. Студентам запрещается передвигать компьютерную технику и оборудование без разрешения преподавателя.
3. Во время занятий запрещается ходить по аудитории, посещать комнаты не технического назначения и громко разговаривать, тем самым отвлекая других учащихся.
4. Выполнять следует только указанные преподавателем задачи. Категорически запрещено выполнять другие работы.
5. Запрещено самостоятельно перемещать аппаратуру.
6. Запрещено запускать игровые и другие сторонние программы или приложения.
7. В случае возникновения неполадок нужно сообщить преподавателю.
8. Студенты не должны пытаться самостоятельно отрегулировать аппаратуру или устранять в ней неполадки.

Требования безопасности после окончания работы

1. После окончания занятия на рабочем месте запрещается оставлять включенной технику и лишние предметы (методические материалы, личные вещи и пр.).

Требования безопасности в аварийных ситуациях

2. При появлении необычного звука или выключения аппаратуры следует немедленно прекратить работу и поставить в известность преподавателя.
3. При появлении запаха гари следует прекратить работу, выключить аппаратуру и сообщить об этом преподавателю. Когда это необходимо, помочь тушить пожар.
4. При попадании человека под напряжение необходимо обесточить соответствующее рабочее место, оказать первую доврачебную помощь и вызвать « скорую».
5. При возникновении пожара необходимо обесточить учебное помещение, вызвать пожарную команду и приступить к тушению пожара средствами, которые есть.
6. В случае несоблюдения студентами требований по охране труда и пожарной безопасности администрация может привлечь их к дисциплинарной и административной ответственности.

ЛІТЕРАТУРА

1. Клочков Д.П., Павлов Д.А. Введение в объектно-ориентированное программирование. / Учебно-методическое пособие. - Изд. Нижегор. ун-та, 1995. - 70с.
2. Уоллес Вонг. Microsoft Visual Basic .NET для "чайников" = Visual Basic.NET For Dummies. — М.: «Диалектика», 2003. — С. 336.
3. Джеффри П. Мак-Манус, Джеки Голдштейн, Кевин Т. Прайс. Обработка баз данных на Visual Basic .NET = Database Access with Visual Basic.NET. — М.: «Вильямс», 2003. — С. 416.
4. Гарбер Г.З. Основы программирования на Visual Basic и VBA в Excel 2007. – М.: СОЛОН-ПРЕССб 2008. – 192 с.: ил. – (Серия "Библиотека студента")
5. Слепцова Л.Д. Программирование на VBA в Microsoft Office 2010. – М.: ООО "И.Д.Вильямс", 2010.– 432 с.: ил.
6. Эффективная работа: Microsoft Office Excel 2003 / М. Додж, К. Стинсон — СПб.: Питер, 2005. — 1088 с: ил.
7. Bill Jelen, Tracy Syrstad. VBA and Macros Microsoft Excel 2010. – Indianapolis, Indiana: QUE, 2010. – 630 с.: ил.
8. Вержбицкий, В. М. Основы численных методов. М., 2002, или Вержбицкий, В. М. Основы численных методов. 2-е изд., перераб. / В. М. Вержбицкий. – М.: Высшая школа, 2005.
9. Воеводин, В. В. Численные методы алгебры. Теория и алгоритмы / В. В. Воеводин. – М.: Наука, 1966.
10. Самарский, А. А. Численные методы / А. А. Самарский, А. В. Гулин. – М.: Наука, 1989.
11. Семушин, И.В. Численные методы алгебры / И.В. Семушин. – Ульяновск: УлГТУ, 2006.
12. Костомаров, Д.П. Вводные лекции по численным методам: Учеб. пособие / Д.П. Костомаров, А.П. Фаворский. – М.: Логос, 2004. – 184 с.
13. Турчак, Л. И. Основы численных методов. 2-е изд, перераб. и доп. М., 2003.
14. Ортега, Дж. Введение в численные методы решения дифференциальных уравнений / Дж. Ортега. – М.: Наука, 1986.