

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт высокоточных систем им. В.П. Грязева

Кафедра «Ракетное вооружение»

Утверждено на заседании кафедры
«Ракетное вооружение»
«16» 01 2019 г., протокол №5

/ Заведующий кафедрой



Н.А. Макаровец

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине (модулю)**

«Информатика»

**основной профессиональной образовательной программы
высшего образования – программы специалитета**

по специальности

24.05.02 Проектирование авиационных и ракетных двигателей

со специализацией

Проектирование ракетных двигателей твердого топлива

Форма обучения: очная

Идентификационный номер образовательной программы: 240502-01-19

Тула 2019 год

Разработчик методических указаний

Дунаев В.А., профессор, д.т.н., профессор

(ФИО, должность, ученая степень, ученое звание)



(подпись)

СОДЕРЖАНИЕ

Введение.....	4
Лабораторная работа № 1	6
Лабораторная работа № 2.....	28
Лабораторная работа № 3.....	44
Лабораторная работа № 4.....	44
Лабораторная работа № 5	49
Лабораторная работа № 6.....	58
Лабораторная работа № 7	66
Лабораторная работа № 8	72
Лабораторная работа № 9	76
Лабораторная работа № 10.....	80
Библиографический список	83

ВВЕДЕНИЕ

Важнейшей особенностью современного этапа научно технического прогресса является стремительно возрастающая роль ЭВМ во всех областях инженерной деятельности - от систем автоматизированного проектирования и управления до контроля технологическими процессами. В настоящее время ЭВМ стали необходимым оборудованием НИИ, конструкторских бюро, заводов. Это позволило от простейших расчетов и оценок различных конструкций или процессов перейти к новому уровню инженерной деятельности - детальному математическому моделированию (вычислительному эксперименту), которое существенно сокращает потребность в натурных экспериментах, а в ряде случаев может их заменить. В связи с этим современный специалист с высшим образованием должен обладать не только высоким уровнем подготовки по профилю своей специальности, но и хорошо знать математические методы решения инженерных задач, ориентироваться на использование вычислительной техники, практически освоить принципы работы на ЭВМ.

Язык Си - один из наиболее популярных современных языков программирования. С момента появления (начало 70-х годов) до настоящего времени он прошел несколько этапов своего развития и совершенствования (в 1983 г. создан ANSI стандарт С). С 1980 года он включает новый подход к решению задач на ЭВМ, названный объектно-ориентированным программированием.

Язык программирования С++ очень популярен. Это объясняется не только популярностью его предшественника С, но и благодаря ориентации на абстракцию данных и объектную ориентированность. Развитие С++ было отмечено активным диалогом сообщества пользователей, которые плодотворным обменом идеями помогли его формировать. В результате, язык С++ приобрел множество черт, которые поддерживают различные методы и приемы программирования.

Язык С++ лучше всего изучать, рассматривая, каким образом взаимодействуют черты языка при написании программ. Как и большинство пользователей С++, во время становления языка, мы имеем возможность увидеть, почему конкретные элементы языка были развиты и почувствовать в их усовершенствовании через их применение и использование.

Выпускник ВУЗа должен уметь правильно и рационально составить программу расчета, используя существующие стандартные модули, отладить ее и правильно интерпретировать полученные результаты.

Лабораторные занятия, предлагаемые в данном сборнике, выполняются на этапе общей базовой подготовки. Они предназначены для закрепления на практике знаний, даваемых в курсе лекций и выработки у студентов практических навыков в программировании, а также развитие навыков оформления разработанных программ.

Каждая из лабораторных работ представляет собой разработку небольшой программы, и предполагает самостоятельное и законченное исследование, включающее составление и отладку программы, анализ результатов расчетов на ЭВМ.

В сборнике содержаться основные сведения теоретического характера, даны конкретные примеры, что вполне достаточно для разработки программ реальных задач. При необходимости следует воспользоваться указанной литературой и курсом лекций.

Единая методическая основа и доступность изложения материала позволяет студентам самостоятельно выполнять предлагаемые работы.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ РАБОТ

Каждая лабораторная работа должна выполняться после изучения студентами соответствующих разделов теоретического курса лекций.

К дальнейшей работе на ЭВМ допускаются только те студенты, которые в процессе выполнения данных лабораторных работ показали знание теоретических вопросов. Работа считается выполненной после того, как преподаватель, проверив результаты и ответы студентов на контрольные вопросы, подписывает предоставленный студентом отчет по установленной форме.

ЛАБОРАТОРНАЯ РАБОТА № 1

РАБОТА В СРЕДЕ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ

1. Цель и задачи работы

При выполнении работы студенты знакомятся с основными элементами среды визуального программирования алгоритмического языка C++, с правилами создания проектов, визуальными компонентами, их свойствами, методами и событиями, получают навыки в написании программ.

Студентам необходимо самостоятельно сформировать на C++ ряд имен переменных, числовых констант и арифметических выражений.

Основы теории

C++Builder предназначен для быстрой разработки приложений (RAD), построенных на современном фундаменте объектно-ориентированного программирования (ООП). C++Builder сам постепенно будет помогать вам овладевать премудростями RAD, ООП и языка C++, поначалу требуя лишь минимальных предварительных знаний. Ваши навыки будут расширяться по мере роста сложности ваших разработок: чем сложнее задача, тем больший по объему код потребуется написать для ее реализации. Получение знаний перестает быть самоцелью. Вы сами убедитесь в том, что C++Builder в корне меняет процесс разработки насколько легче и быстрее вы сможете получать работающие и надежные программы для операционных систем Windows, чем при использовании традиционных интерфейсных оболочек других систем.

Иллюстрировать мощность и гибкость C++Builder будем на примере построения тестовых приложений, развивая их от простейших прототипов до законченных рабочих программ.

Первое знакомство

Как начинающие, так и опытные программисты начинают знакомство с новой системой с попытки создать простую программу, а чтение документации откладывается до лучших времен. При этом оценивают разные аспекты разработки: насколько полезными оказываются ваши интуиция и опыт, лаконичность и объем кода, достоинства сервиса среды, временные затраты, удобства отладки и многое-многое другое. Последуем и мы по этому пути знакомства со средой программирования C++Builder.

Мы получим общее представление о работе со следующими основными инструментами интегрированной среды:

- Палитра компонент содержит более 100 повторно используемых компонент, предлагаемых для построения приложений.
- Редактор форм предназначен для создания интерфейса программы с пользователем.
- Редактор кода предназначен для написания текста программы, в частности, функций обработки событий.
- Инспектор объектов позволяет визуально устанавливать свойства объектов без необходимости рутинного программирования и содержит события, которые можно связывать с кодами реакции объектов на их возникновение.

- Хранилище объектов содержит такие объекты, как формы и модули данных, которые разделяются многими приложениями с целью уменьшения временных затрат при разработке.

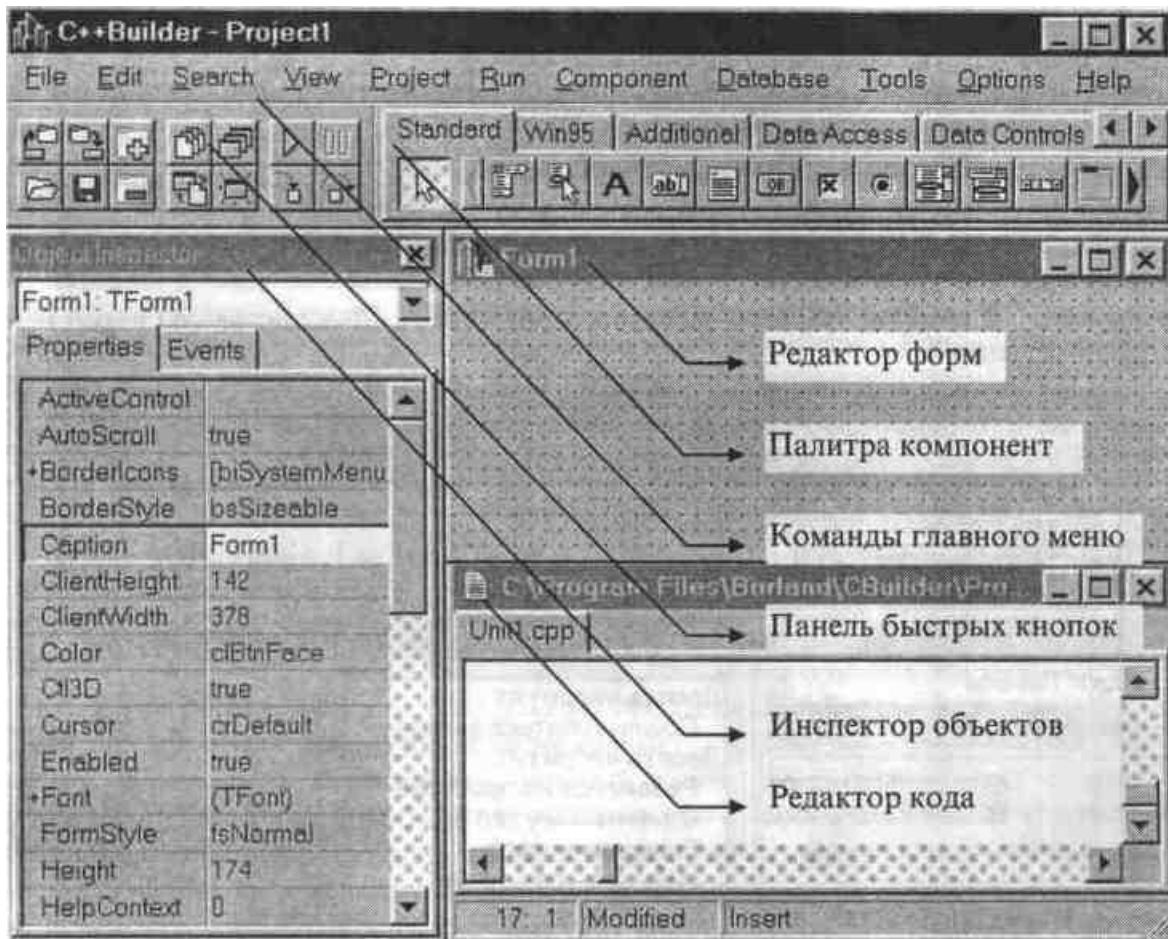


Рис. 1. Основные визуальные инструменты интегрированной среды C++ Builder.

C++Builder реализует визуальную методику построения приложений посредством выбора из Палитры компонент нужных управляющих элементов (рис. 1). С каждой компонентой (например, кнопкой) связаны свойства, которые меняют ее вид и поведение. Любая компонента может вызывать серию событий, которые определяют ее реакцию на различные воздействия. В дальнейшем изложении символы => обозначают те действия, которые вы будете совершать в среде C++Builder.

=> Вызовите C++Builder и начните работу над новым приложением по команде File | New Application из главного меню.

=> Щелкайте мышью по вкладкам Палитры компонент, просмотрите имеющийся ассортимент элементов интерфейса программы с пользователем.

Переходя от вкладки к вкладке Палитры, можно заметить, что набор доступных компонент меняется. Когда курсор мыши останавливается на значке компоненты, возникает подсказка с ее названием. Если нажать клавишу F1, справочная служба системы выдаст полную информацию о выбранной компоненте.

Визуальное проектирование

Наше первое приложение будет генерировать решение квадратного уравнения. В начальной версии потребуется только 12 объектов: 5 полей редактирования, 5 меток и

две кнопки. Перенесем компоненты на форму проектирования и начнем постепенно развивать приложение. Метод перетаскивания (drag-and-drop) состоит в следующем: нажмите кнопку мыши на выбранной компоненте, переведите курсор на любое место формы, а затем снова нажмите кнопку мыши. Для начала ограничимся "стандартными" компонентами Палитры:

- => Выберите вкладку Standard.
- => Перетащите компоненты поля редактируемого ввода TEdit.
- => Перетащите компоненты TLabel.
- => Перетащите компоненты TButton.
- => Расположите компоненты и измените их размеры так, как вы хотели бы их видеть в окне вашего приложения.

С помощью Инспектора объектов определите начальные значения свойств компонент.

Теперь можно переключиться на Редактор кода и написать, как было принято, любую программу на языке C++, включая последние расширения стандарта ANSI/ISO. Однако, попытаемся сначала воспользоваться новыми средствами быстрой разработки приложений и дополнительными атрибутами компонент, заложенными в C++Builder.

Свойства, методы и события

Быстрая разработка приложений подразумевает поддержку свойств, методов и событий компонент в рамках объектно-ориентированного программирования. Свойства позволяют вам легко устанавливать разнообразные характеристики компонент, такие как названия, контекстные подсказки или источники данных. Методы (функции-члены) производят определенные операции над компонентным объектом, в том числе и такие сложные как воспроизведение или перемотка устройства мультимедиа. События связывают воздействия пользователя на компоненты, такие как активизация, нажатие кнопок или редактируемый ввод - с вашими кодами реакции на эти воздействия. Кроме того, события могут возникать при таких специфических изменениях состояния компонент как обновление данных в интерфейсных элементах доступа к базам данных. Работая совместно, свойства, методы и события образуют среду RAD интуитивного программирования надежных приложений для Windows.

=> В Инспекторе объектов укажите вкладку Событий (Events), чтобы увидеть все события, ассоциированные с выбранным объектом.

=> Дважды щелкните мышью по компоненте кнопки, которую вы поместили на форму.

=> В открывшемся окне Редактора кода курсор покажет позицию для ввода инструкций в тело функции Button1Click, предназначеннной для обработки события OnClick, возникающего при нажатии кнопки.

Этап проектирования первой версии приложения на этом завершается и можно приступить к созданию рабочей программы.

=> Командой главного меню Run | Run запустите процесс компиляции и сборки приложения.

=> После вызова программы несколько раз нажмите на кнопку "Результат".

Технология двунаправленной разработки

C++Builder не ставит никаких барьеров между программистом и его кодом. Технология двунаправленной разработки Two-Way Tools обеспечивает контроль за вашим кодом посредством гибкого, интегрированного и синхронизированного взаимодействия между инструментами визуального проектирования и Редактором кода.

Чтобы проследить за тем, как действуют инструменты двунаправленной разработки, выполните следующие операции:

=> Откройте контекстное меню Редактора кода щелчком правой кнопки мыши, а затем с помощью опции Swap Cpp/Hdr Files переключитесь на файл объявлений Unit1.h..

=> Организуйте экранное отображение инструментов так, чтобы одновременно видеть проектируемую форму и файл Unit1.h в окне Редактора кода.

=> Перетащите еще одну компоненту кнопки OK Button на форму. В свойстве Caption кнопки укажите ее смысловое название.

Последите за тем, что как только вы перенесли кнопку на форму, объявление объекта Button2 моментально появится в файле Unit1.h, а определение события OnClick генерирует объявление метода Button2Click обработчика этого события (Рис. 2). Такая синхронизация процессов проектирования формы и автоматической генерации кода действительно ускоряет визуальную разработку C++ приложения, полностью сохраняя контроль над исходным текстом программы.

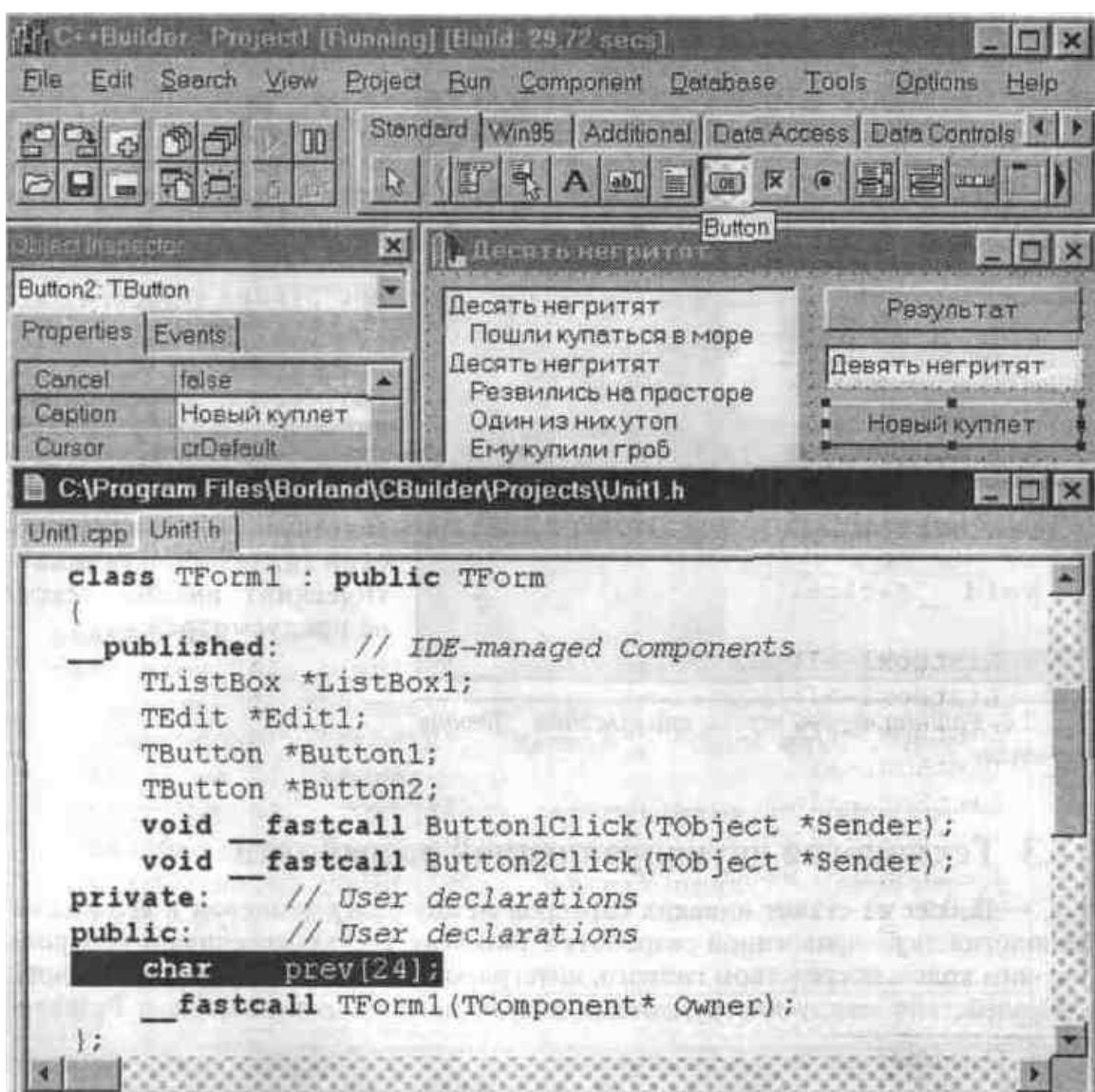


Рис. 2. C++Builder автоматически генерирует объявления в файле Unit1.h, поддерживая постоянный контроль программиста над его кодом

Следует запомнить, что C++Builder ассоциирует с каждым приложением три исходных файла со следующими именами по умолчанию:

- Unitl.cpp хранит исполняемый код реализации вашего приложения. Именно в нем вы записываете обработчики событий, отвечающие за реакцию программы при воздействии пользователя на объекты компонент.

- Unitl.h содержит объявления всех объектов и их конструкторов. Обратите внимание на ключевое слово `_fastcall` в объявлениях функций обработки событий, которые C++Builder генерирует автоматически. Благодаря `_fastcall` передача параметров организуется не через стек, а через регистры центрального процессора. Вызовы обработчиков событий происходят очень часто, поэтому экономия времени, затрачиваемого на выборку параметров из памяти стека, оказывается весьма ощутимой. Здесь кроется одна из причин высокого быстродействия приложений, которые компилирует и собирает C++Builder.

- Projectl.cpp обслуживает все объекты, заключенные в приложении. Любая новая форма, программный модуль или модуль данных автоматически включаются в проектный файл. Вы можете просмотреть в окне Редактора кода содержание исходного текста проектного файла с помощью команды главного меню View | Project Source или выбрав одноименную опцию из контекстного меню Администратора проекта. Ни в коем случае не редактируйте проектный файл вручную!

Быть может, завершив разработку первого приложения, вы захотите сохранить исходные файлы для следующего сеанса, выполнив одно из следующих действий:

=> Команда File | Save All сохраняет все исходные файлы приложения.

Команда File | Save сохраняет оба файла программного модуля, а команда File | Save As позволяет дать им новое имя.

Команда File | Save Project As сохраняет изменения всех составляющих проектного файла, используя текущие имена файлов.

Использование проектных шаблонов

Использование готовых проектных шаблонов из Хранилища объектов дает вам возможность начать разработку, опуская типичные для многих приложений предварительные операции по составлению главного меню и панели кнопок быстрого вызова, организации стандартных диалогов вызова и сохранения файлов. Внесенные вами изменения никак не отражаются на использовании того же проектного шаблона другими разработчиками.

Чтобы создать на основе проектного шаблона прототип приложения для работы в режиме многодокументного интерфейса (MDI), выберите из Хранилища объектов MDI Application.

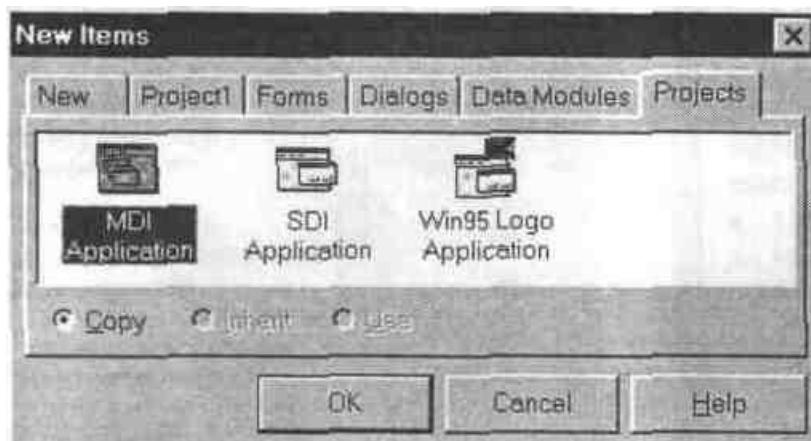


Рис. 3. Проектные шаблоны.

Рис. 4 показывает главную форму прототипа приложения, спроектированную на базе готового шаблона.

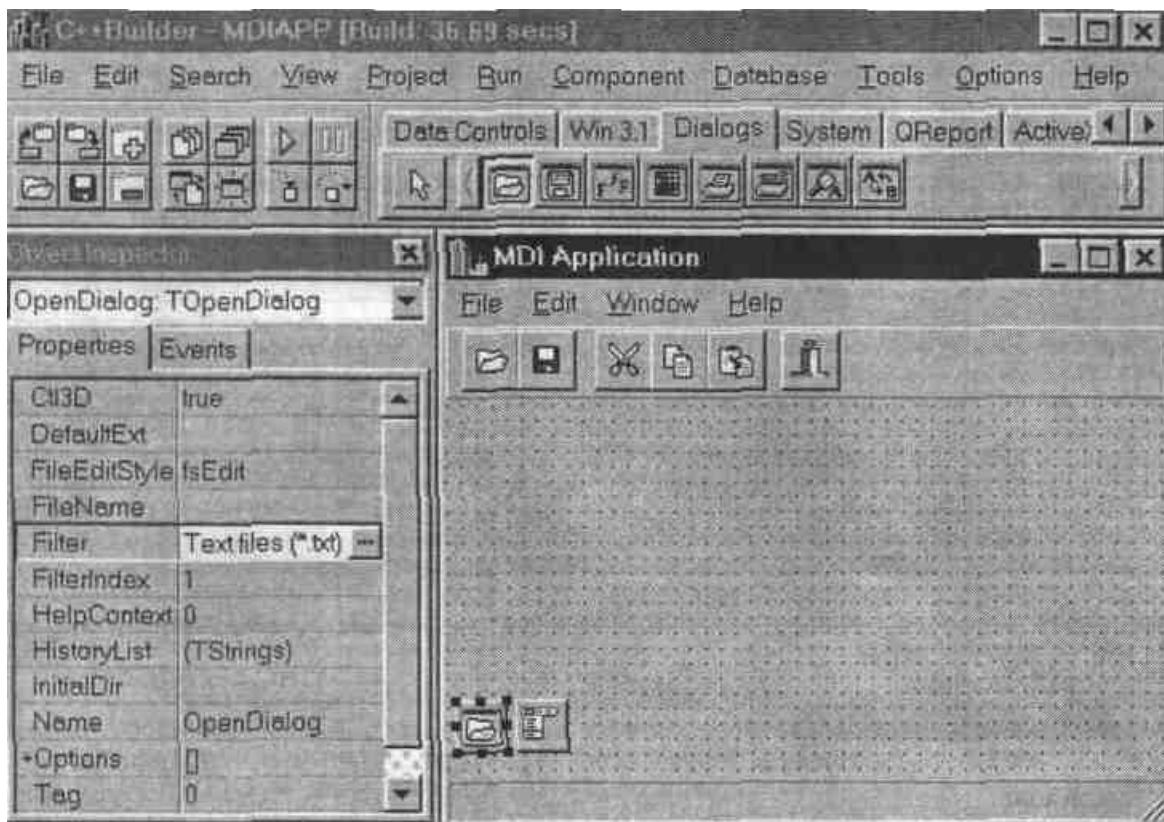


Рис. 4. Главная форма прототипа MDJ приложении для работы с текстовыми файлами.

Если вы скомпилируете и соберете такое приложение, то увидите, что оно "умеет" только оперировать с окнами в режиме MDI и вызывать диалог открытия файлов, не заполняя окна текстовым содержимым выбранных файлов. То есть прототип оказался нефункциональным и практически бесполезным. Чтобы придать приложению некоторое осмысленное поведение, необходимо добавить на форму необходимые для конкретного приложения компоненты.

Палитра компонент - краткий обзор

Компоненты вкладки Standard

Компоненты этой вкладки осуществляют включение в ваше приложение следующих типовых интерфейсных элементов Windows:

TManMeni	Создает панель команд главного меню для формы.
TPopUpMerm	Создает "выскакивающее" меню для формы или для другой компоненты.
TLabel	Отображает на форме текст названия, который нельзя редактировать.
TEdit	Отображает область редактируемого ввода одиночной строки информации на форме.

TButton	Создает кнопку с надписью.
TCheckBox	Создает элемент управления с двумя состояниями.
TRadioButton	Создает элемент управления с двумя состояниями.
TListBox	Отображает область списка текстовых строк.
TComboBox	Создает комбинацию области редактирования и выпадающего списка текстовых строк.
TScrollBar	Создает линейку прокрутки для просмотра содержимого окна, формы, списка или диапазона значений.
TGroupBox	Создает контейнер, объединяющий на форме логически связанный группу некоторых компонент.
TRadioGroup	Создает контейнер, объединяющий на форме группу логически взаимоисключающих радио-кнопок.
TPanel	Создает панель инструментов или строк состояния.

Компоненты вкладки Windows

Компоненты этой вкладки осуществляют включение в ваше приложение следующих типовых интерфейсных элементов Windows:

TTabControl	Отображает набор полей, имеющих вид частично перекрывающих друг друга картотечных вкладок.
TPageControl	Отображает набор полей, имеющих вид частично перекрывающих друг друга картотечных вкладок, для организации многостраничного диалога.
TTreeView	Отображает древовидный перечень элементов - заголовков документов, записей в указателе, файлов или каталогов на диске.
TListView	Отображает древовидный перечень элементов в различных видах - по столбцам с заголовками, вертикально, горизонтально, с пиктограммами.
TImageList	Создает контейнер для группы изображений.
THeaderControl	Создает контейнер для заголовков столбцов.
TRichEdit	Отображает область редактируемого ввода множественных строк информации в формате RTF.

TStatusBar	Создает строку панелей состояния для отображения статусной информации.
TTrackBar	Создает шкалу с метками и регулятором текущего положения.
TProgressBar	Создает индикатор процесса выполнения некоторой процедуры в приложении.
TUpDown	Создает спаренные кнопки со стрелками "вверх" и "вниз". Нажатие этих кнопок вызывает увеличение или уменьшение значения свойства Position.
THotKey	Используется для установки клавиш быстрого вызова во время выполнения программы.

Компоненты вкладки Additional

Компоненты этой вкладки осуществляют включение в ваше приложение следующих специализированных интерфейсных элементов Windows:

TBitBtn	Создает кнопку с изображением битового образа.
TSpeedButton	Создает графическую кнопку быстрого вызова.
TMaskEdit	Создает область редактируемого ввода данных специфического формата.
TStringGrid	Создает сетку для отображения строк по строкам или столбцам.
TDrawGrid	Создает сетку для отображения графических данных по строкам или столбцам.
TImage	Создает на форме контейнер для отображения битового образа, пиктограммы или метафайла.
TShape	Рисует простые геометрические фигуры.
TBevel	Создает линии и рамки с объемным видом.
TScrollBox	Создает контейнер переменного размера с линейками прокрутки, если это необходимо

Основы программирования

Вначале каждой программы находится раздел описания используемых библиотек, за тем по необходимости используется раздел описания глобальных переменных, затем располагаются функции, из которых состоит программа. Функции могут располагаться в произвольном, удобном для пользователя порядке. Под функцией или

процедурой понимается законченная последовательность арифметических или логических действий многократно выполняющихся в программе. Результат функции или процедуры передается в вызывающую точку либо с помощью формальных параметров, либо с помощью имени функции, либо с помощью глобальной переменной. Тело функции заключается между фигурными скобками { }.

1. Константы

В языке C++ существуют три типа констант:

целые 23, 456, - 234

дробные 123.4, 567.987, - 455.987

символьные 'A', '2'

Для описания целых чисел в C++ существуют следующие ключевые слова:

short int (диапазон возможных значений $\sim \pm 128$)

int (диапазон возможных значений $\sim \pm 32\ 000$)

long int (диапазон возможных значений $\sim \pm 2\ 147\ 483\ 648$)

Для описания дробных чисел существуют ключевые слова:

float (диапазон возможных значений $\sim \pm 3.4E\pm 38$)

double (диапазон возможных значений $\sim \pm 3.4E\pm 38$)

long double (диапазон возможных значений $\sim \pm 1.7E\pm 308$)

Переменные и наименования переменных

Термин “переменная” употребляют в C++ для обозначения величины, обращение к которой производится через ее наименование и которая может принимать различные значения, а не ограничена каким-либо одним.

Имена переменных:

имя начинается с буквы или с символа “ _ ”;

в имени не может употребляться “ - ”;

имя отражает сущность переменной;

в имени принято каждое новое слово писать с большой буквы;

заглавные и прописные буквы различаются.

Операции и выражения

В C++ существует четыре основных операции, каждая из которых обозначается особым символом:

сложение +

вычитание -

умножение *

деление /

Операции присваивания.

В математике и $A=2.8$ и $X+2=Y+1$ являются верными, (1-присваивание, 2-равенство). В C++ существуют две отдельные операции: присваивание и сравнение на равенство:

- 1) $A=2.8$ - присваивание, в ячейку памяти A помещают число 2.8, при этом старое содержимое ячейки A стирается;
- 2) $X==Y-1$ - операция сравнения, используется в логических операторах.

Следует помнить, следующие правила написания программ. Два символа операций не должны стоять рядом. Поэтому a^*-b является бессмысленным выражением, но $a^*(-b)$ - допустимо. Скобки используются для указания очередности выполнения операций так же, как и в обычной математической записи.

Ниже показано, что C++ позволяет в операции присваивания можно отметить двукратное обращение к переменной $X *= 2$, $X /= 2$, $X += 2$, $X -= 2$, что позволяет ускорить процесс вычисления.

В алгоритмическом языке C++ определены также следующие операции изменения переменной на 1:

- | | |
|--|---|
| $\begin{cases} i++ \\ i-- \end{cases}$ | - вначале использует значение i, а затем изменяет на единицу. |
| $\begin{cases} ++i \\ --i \end{cases}$ | - вначале изменяет на единицу, а затем использует значение i. |

Математические функции

В C++ предусмотрена возможность использования некоторых элементарных математических функций, таких, как логарифм, экспонента, синус, косинус, и т.д. Конкретный набор функций, предусмотренных языке зависит от варианта C++, однако все варианты предусматривают возможность вычисления нижеперечисленных функций и ряда других.

Математическая функция	Наименование
Квадратный корень	$\text{sqrt}(x)$
Экспонента	$\text{exp}(x)$
Синус (рад)	$\text{sin}(x)$
Косинус (рад)	$\text{cos}(x)$
Арктангенс, угол в радианах	$\text{atan}(x)$
Натуральный логарифм	$\text{log}(x)$
Абсолютная величина	$\text{abs}(x)$
Возведение x в степень y	$\text{pow}(x,y)$

Угол должен быть выражен в радианах. Например, необходимо вычислить косинус угла X. Запись в каком-либо выражении $\text{cos}(x)$ приведет к вычислению косинуса угла X.

Основные операторы

Арифметические операторы

Основные элементы языка C++, рассмотренные ранее, широко применяются при составлении исходных программ. Наиболее важным применением является вычисление нового значения переменной, которое производится с помощью арифметического оператора присваивания.

Присваивание, как например в выражении `ratio=a/b` или `ch=getch()`, является одной из основных операций. В языке C++ присваивание обозначается знаком равенства `=`; значение с правой стороны присваивается переменной с левой стороны.

Знак равенства в арифметическом операторе присваивания используется не совсем в том смысле, что в обычной математической записи. Значение знака равенства в этом случае будет следующим: заменить прежнее значение переменной, стоящей в левой части, на значение выражения, стоящего в правой части оператора. Таким образом, оператор `A = B + C` есть ни что иное, как команда вычислить сумму значений переменных `B` и `C` и заменить прежнее значение переменной `A` этой суммой. Прежнее значение переменной при этом теряется, но значения переменных `B` и `C` остаются неизменными.

В другом примере арифметического оператора присваивания эти специфические особенности знака равенства проявляются предельно ярко. Оператор типа `N=N+1` имеет смысл: присвоить переменной `N` ее старое значение плюс 1. Этот тип оператора, явно не имеющий аналогии в обычной математической записи, широко распространен в C++.

Присваивания можно связывать таким образом: `sum=a=b`. В таком случае вычисление производится справа налево, т.е. значение `b` будет присвоено переменной `a`, а затем присвоено переменной `sum`, при этом все три переменные получат одно и то же значение (первоначальное значение переменной `b`).

Несколько примеров помогут уяснить применение арифметических операторов присваивания. Предположим, что `A`, `B`, `C`, `D` и `X` уже вычислены с помощью предыдущих операторов, и теперь необходимо вычислить значение `R` из соотношения:

$$R=A+BX/C+DX$$

Необходимые вычисления могут быть произведены с помощью следующего оператора:

$$r=(a+b*x)/(c+d*x)$$

Ни одно из значений переменных в правой части не будет изменено после выполнения оператора, переменной `r` будет присвоено новое значение, старое значение `r` будет утеряно.

Предположим, что необходимо вычислить один из корней квадратного уравнения $aX^2 + bX + c = 0$. По прежнему `a`, `b`, и `c` должны быть вычислены с помощью предыдущих операторов или введены с

помощью оператора ввода. Следующий оператор определит необходимый порядок вычислений и присвоит вычисленное значение переменной $x1$:

$$x1=(-b+\sqrt{b^2-4.0*a*c})/(2.0*a).$$

Было бы полезно еще раз напомнить о значении скобки.

Скобки, в которые заключено выражение $b^2-4.0*a*c$, необходимы для того, чтобы обозначить аргумент функции квадратного корня. Скобки вокруг числителя необходимы для того, чтобы все, что находится перед знаком деления, было разделено на то, что следует далее. Скобки вокруг выражения $2.0*a$ показывают, что это выражение находится в знаменателе: если бы эти скобки отсутствовали, то числитель был бы разделен на 2, а результат деления был бы умножен на A.

Последний пример показывает, как аргумент функции, в случае необходимости, может сам быть другой функцией. Предположим, что необходимо вычислить значение V из формулы:

$$V=1/\cos X+\log[\tan X/2].$$

Оператор для вычисления V запишется следующим образом:

$$v=(1.0/\cos(x))+\log(\tan(x/2.0))).$$

Вполне допустимо введение промежуточных переменных, что может облегчить чтение исходной программы;

$$y=\text{abs}(\tan(2.0)); z=1.0/\cos(x); v=z+\log(y).$$

Примеры, приведенные ниже, показывают правильно написанные арифметические операторы и эквивалентные им математические формулы (табл.1).

Таблица 1

Арифметический оператор	Исходная формула
$b=-1.0/(2.0*x)+a*a/(4.*x*x);$	$b=-1/2x+a^2/4*x^2$
$Fy=x*(x*x-y*y)/(x*x+y*y);$	$Fy=x*(x^2-y^2)/x^2+y^2$
$C=1.112*D*r1*r2/(r1-r2);$	$C=1.112*D*r1*r2/(r1-r2)$
$Y=(1.E-6+A*X*X*X)**(2.0/3.0);$	$Y=(1.E-6+A*X^3)**(2./3.)$
$J=4*k-6*k1*k2;$	$J=4*k-6k_1*k_2$
$I=I+1;$	$I_{\text{нов.}}=I_{\text{стар.}}+1$
$K=12;$	$k=12$
$Pi=3.1415927;$	$\pi=3.1415927$
$M=2*M+10*J;$	$M_{\text{нов.}}=M_{\text{стар.}}+10*J$
$R=\cos(x)+X*\sin(x);$	$R=\cos x+x*\sin x$
$S=-\cos(x)/4.0;$	$S=-\cos x/4$
$T=\text{atan}(1.414214*tan(x));$	$T=\arctg(1.414214 \operatorname{tg} x)$

Наименования переменных выбраны произвольно; любые другие допустимые наименования переменных можно было бы употребить с тем же успехом. Кроме того, подразумевается, что все величины переменных в правой части известны к моменту начала действий арифметического оператора.

Приведенные примеры подчеркивают необходимость строгого соблюдения правил написания выражений, так как успешная работа транслятора C++ невозможна при нарушении этих правил. Каждый оператор, приведенный ниже, содержит по крайней мере одну ошибку (табл. 2).

Таблица 2

Арифметический оператор	Ошибка
$Y=2.0X+A;$	Отсутствует * (знак умножения)
$3.14=X-A;$	Слева должно стоять наименование переменной
$Y=((2*X-5)+6;$	Неодинаковое количество правых и левых скобок
$PI=3,1415927;$	Запятые в константах употреблять нельзя
$Z=a+*g-3;$	Нельзя ставить два и более знаков операций подряд
$\text{sqrt}(z)=z*8/5.5;$	Имя функции <code>sqrt()</code> нельзя использовать в качестве имени переменной

Передача управления. Операторы перехода

Элементы языка C++, рассмотренные ранее, позволяют производить весьма сложные вычисления, но при использовании их остается не снятым серьезное ограничение. Все операторы могут выполняться только в той последовательности, в которой они написаны, и ни в какой другой. Снимают это ограничение операторы перехода.

В языке "C" предусмотрен оператор `goto` и метки для ветвления. С формальной точки зрения оператор `goto` никогда не является необходимым, и на практике почти всегда можно обойтись без него. Тем не менее, существуют ситуации, где оператор `goto` может найти свое место и обеспечить наиболее оптимальное решение. В связи с этим этот оператор сохранен даже в языке C#. Метка имеет такую же форму, что и имя переменной, и за ней всегда следует двоеточие. Метка может быть приписана к любому оператору той же функции, в которой находится оператор `goto`.

Оператор `goto` определяет передачу управления какому-либо оператору, отличному от следующего. Этот оператор записывается в виде `goto met;`, где `met` - метка, на которую осуществляется переход при исполнении программы. Когда в ходе выполнения программы очередь доходит до оператора `goto met;`, то следующим будет исполнен оператор помеченный меткой `met`. Этот оператор может быть расположен как до, так и после оператора `goto`. Оператор `goto met;` является оператором безусловного перехода.

Однако часто требуется изменять последовательность выполнения операторов в зависимости от того, что происходит в процессе выполнения программы.

Одним из таких средств, позволяющих решить эту проблему, является логический оператор `if`. Инструкция `if` используется в тех случаях, когда

необходимо решить, должна ли быть выполнена конкретная инструкция программы (if по английски значит "если"). Структура if выглядит следующим образом:

```
if (condition) instruction;
```

Этой записью мы говорим: "Если некоторое условие выполняется (является истинным), инструкция должна быть выполнена". То есть, компьютер, встретив ключевое слово if, выполняет инструкцию, следующую за ним, если условие в скобках является истинным.

Оператор if выполняет только одну инструкцию. Если возникает необходимость, чтобы при выполнении одного условия выполнялось несколько команд, следует использовать составную инструкцию. Составной инструкцией называется последовательность любых инструкций, заключенных в фигурные скобки. С точки зрения синтаксиса языка такая последовательность будет рассматриваться как единая инструкция.

В общем случае вид этого оператора следующий:

```
if (логическое выражение)
{
    действия истинности оператора;
}
else
{
    действия "иначе";
}
```

Блок "истина" или блок "иначе" может отсутствовать. Если в программе встречается несколько операторов else, то каждый оператор else относится к ближайшему оператору if.

Наряду с логическим оператором используются логические выражения:

```
a < b - а меньше b
a > b - а больше b
a<=b - а меньше или равно b
a>=b - а больше или равно b
a==b - а равно b
a !=b - а не равно b
```

Логическое "и" объединяет несколько логических выражений $((a < x) \&\& (x < b))$ эквивалентно $a < x < b$.

Логическое "или" $((a > x) || (x > b))$ если одно из двух условий верное, то оно и выполняется.

Сама по себе инструкция if используется в тех случаях, когда важно выполнить некие действия при истинности определенного условия. Возможна ситуация, когда при истинности условия должен быть выполнен один набор инструкций, а в противном случае - другой. Допустим, мы хотим, чтобы в тех случаях, когда цена какого-либо товара ниже той, для

которой установлен налог на предметы роскоши, программа генерировала сообщение: "Для данного наименования налог не установлен". Это можно сделать с помощью двух инструкций if.

```
if (cost > 40000.00)
{ luxury= cost * 0.005;
printf("Размер налога для данного наименования составляет %f",
luxury);
}
if (cost < 40000.00)
puts("Для данного наименования налог не установлен");
```

Но есть более эффективный способ. Можно объединить обе инструкции в одну, пользуясь тем, что есть только два возможных случая в использовании одной и той же переменной: либо цена товара больше 40 тысяч долларов, либо цена товара меньше или равна указанной сумме. Если одно из условий не выполняется, следовательно, выполняется второе условие, так что можно скомбинировать их, используя ключевое слово else (которое переводится как "иначе").

```
if (condition) instruction1;
else           instruction2;
```

Здесь сказано: "Если условие истинное, то должна быть выполнена команда, являющаяся частью инструкции if, иначе надо выполнить инструкцию, следующую за else". Инструкция, помещенная после ключевого слова else, выполняется только в том случае, если условие оказалось ложным. Если возникает необходимость выполнить в этом случае несколько инструкций, можно использовать составную инструкцию, заключив ее в фигурные скобки точно так же, как для if. Точка с запятой ставится в конце каждой инструкции и не ставится после ключевого слова else.

Конструкция if...else может содержать инструкции любого типа. Они могут включать ввод и вывод значений, выполнение математических операций или вызов собственных функций. Но инструкция в условии может оказаться и другой инструкцией if. В этом случае она будет называться вложенной инструкцией. Ниже приведен пример, где одна инструкция if вложена в другую:

```
if (income > 100000)
if (status == 'S') taxrate = 0.35;
```

Второе условие проверяется только в том случае, если выполнено первое, так что значение переменной taxrate присваивается только при выполнении обоих условий. Ту же самую логическую конструкцию можно было записать следующим образом:

```
if (income > 100000 && status == 'S') taxrate = 0.35;
```

Обе инструкции выполняют одну и ту же задачу, но второй способ записи, с использованием оператора &&, кажется более ясным, так как нет

необходимости расшифровывать смысл второй инструкции if. Достаточно просто прочитать инструкцию, чтобы понять принцип действия:

"Если значение переменной income больше чем 100000 и одновременно переменная status имеет значение "S", переменной taxrate присваивается значение 0.35".

Как правило, любые две последовательно вложенные инструкции if можно заменить одной инструкцией, использующей логический оператор "И". Опять же, как правило, имеет смысл избегать вложенных инструкций if, так как они могут приводить к появлению запутанных ситуаций, логических ошибок и трудных для чтения фрагментов текста программы. Посмотрите на следующий пример:

```
float income;
cin>>income;
if (income > 20000.00)
    if (income < 100000.00) cout<<"Размер налога составляет 22%";
else cout<<"Доход меньше 20000$ - налог равен 15%";
```

В этом случае логика рассуждений автора программы была, по-видимому, такова: "Если значение переменной income больше 20000 и меньше 100000, следует вывести одно сообщение, а если значение income меньше 20000, должно быть выведено второе сообщение".

Компиляция программы пройдет без ошибок, но, к несчастью, работать она будет неправильно. Когда вы введете значение меньшее, чем 20000, не будет выполняться ни одна из функций cout<<, а когда введете значение больше 100000, подоходный налог окажется определенным в размере 15 процентов.

Причина ошибки кроется в том, что ключевое слово else связано с ближайшей к нему инструкцией if независимо от отступов, которые были сделаны в тексте программы. В этой программе использование табуляции создает впечатление, что else связано с первым условием if, но это не так. На самом деле else связано со второй инструкцией if, которая выполняется только в тех случаях, когда значение переменной income больше 20000 и меньше 100000.

Если вы хотите, чтобы программа работала в соответствии с вашей логикой, таким образом :

```
if (income >= 20000.00)
{ if (income <=100000.00)cout<<"Размер налога составляет 22%";
}else cout<<"Доход менее 20000$ - налог равен 15%";
```

Фигурные скобки изолируют вложенную инструкцию, и теперь ключевое слово else действительно будет связано с первой инструкцией if.

Еще лучше написать эту же программу с использованием только одной инструкции if.

```
if (income >= 20000.00 && income <= 100000.00)
    cout<<"Размер налога составляет 22%";
else cout<<"Доход не превышает 20000$, налог составляет 15%";
```

В этом варианте мы полностью исключаем путаницу между вложенными инструкциями if и использование дополнительных наборов фигурных скобок.

Даже теперь, после того как мы разобрались с логикой программы, она все еще имеет некоторое упущение. Все сообщения, представляемые программой, относятся к уровню дохода, не превышающему 100 тысяч долларов. В хорошо продуманной программе должны быть приняты во внимание все возможные ситуации. Если в программу, в том виде, в каком она существует сейчас, ввести значение переменной income, равное 150000, программа не выдаст никакого сообщения. Значит ли это, что и подоходный налог платить не обязательно?

Один из способов учета всех возможных вариантов в использовании вложенных инструкций if...else.

Если в программе следует учесть больше трех возможных вариантов, конструкция с вложенными инструкциями if...else может оказаться очень запутанной. В таких случаях в качестве альтернативы используется переключатель switch. Переключатель switch представляет собой структуру, построенную по принципу меню, и содержит все возможные варианты условий и инструкции, которые следует выполнить в каждом конкретном случае.

Оператора switch предназначен для организации выбора из множества различных вариантов.

Формат оператора:

```
switch (n) // заголовок оператора
{
    case a: ... ;
    case c: ... ; break;
    case b: ... ; break;
    .....
    default: ... ;
}
```

Если логический оператор if представляет собой ключ с двумя выходами, то оператор выбора switch представляет собой ключ со многими выходами в зависимости от значения n. n - либо некоторая переменная или выражение, например целого типа, символьная - это возможное значение переменной n или выражения.

Схема выполнения оператора switch:

- 1) Вычисляется n.
- 2) Отыскивается строка case, которая совпадает с n и программа начинает выражаться с данными найденной строки и далее по порядку.
- 3) Если не один из операторов case не совпадает с n, то выполняется строка default. Оператор default может быть опущен.

Если найден вариант case, который совпадает с n и необходимо выполнить только один этот случай и после этого прекратить выполнить строки оператора выбора, необходимо использовать оператор break.

Таким образом, оператор switch дает специальный способ выбора одного из многих вариантов, который заключается в проверке совпадения значения данного выражения с одной из заданных констант в соответствующем ветвлении. Например:

```
answer=getchar();
switch(answer)
{
    case '1': cout<<"Один\n"; break;
    case '2': cout<<"Два\n"; break;
    case '3': cout<<"Три\n"; break;
    case '4': cout<<"Четыре\n"; break;
    default: cout<<"Значение большее 4\n";
}
```

В круглых скобках после переключателя switch находится переменная типа int или char, следом расположен блок инструкций, заключенных в фигурные скобки, которые содержат ряд ветвей case. Каждая ветвь case выполняет инструкции, основываясь на возможном значении переменной. Это значение должно быть представлено в виде целого числа или символа, заключенного в одинарные кавычки, либо в виде имени целочисленной или символьной константы.

Ветвь case '1':, например, предписывает программе выполнить следующие ниже инструкции, если значение переменной переключателя switch соответствует символу '1'. Если значение переменной отличается от единицы, компилятор переходит к проверке условия второй ветви case.

В тех случаях, когда значение переменной удовлетворяет условию ветви case, выполняются инструкции, следующие за данным условием. Инструкция break в конце каждой ветви case передает управление в конец switch, так что, как только выполнены инструкции одной из ветвей case, остальные игнорируются и выполнение switch завершается.

Если значение переменной не удовлетворяет условиям ни одной из ветвей case, выполняется ветвь, помеченная инструкцией default. Это дает возможность учесть все возможные варианты ввода. После инструкции default нет необходимости ставить break, поскольку она всегда является последней в процедуре выполнения switch. Имеет смысл включать default даже тогда, когда вы полагаете, что учли все возможные условия и все возможные случаи ввода значений.

Если вы пропустите инструкцию break, компьютер выполнит все инструкции, помещенные в соответствующей ветви case, и далее, вплоть до первого встреченного в тексте break. Вы можете использовать эту особенность для выполнения определенного набора инструкций, при наличии нескольких равносценных вариантов ответа, например, так:

```
case 'Y':
case 'y': cout<<"Вы ответили \"Да\"\n"; break;
case 'N':
case 'n': cout<<"Вы ответили \"Нет\"\n"; break;
```

Рассмотрим еще один пример: составить функцию, принимая в качестве аргумента номер месяца года, а возвращает название месяца словом.

```
char* mes(int n)
{ char m[20]; // переменная, которая принимает название месяца
  switch (n)
  { case 1: strcpy (m, "январь"); break;
    case 2: strcpy (m, "февраль"); break;
    case 3:
    case 4:
    .....
    case 12: strcpy (m, "декабрь"); break;
    default: strcpy (m, "ошибка");
  }
  return m;
}
```

Массивы.

Очень часто вашей программе может потребоваться сохранить и обрабатывать некоторое множество значений одного и того же типа. Формат определения массива следующий:

тип_данных имя_массива [размер_массива]

Примеры описания массива:

```
int temperature[7]; //Без инициализации
int temperature[7]={78,79,90,96,85,120,55};
```

Для доступа к элементу массива используется имя массива, за которым следует индекс в квадратных скобках. Пример:

```
void func()
{
  int array[10];
  if(условие) array[5]=67;
}
```

Вы можете сделать массив многомерным, задав несколько размеров, заключенных в скобки. Двумерные массивы обычно используются для представления матриц, координат на плоскости. Например:

```
int matrix[5][5];
```

Если размер массива заранее неизвестен, а также для создания очень больших массивов используют динамическое выделение памяти с помощью оператора new:

```
int n, *x;
cin>>n;
x=new int [n];
```

После окончания работы с таким массивом необходимо освободить выделенную память с помощью оператора `delete`:

```
delete[] x;
```

Аналогично можно выделять память для многомерных массивов.

Операторы цикла.

Оператор `while`.

Ключевое слово `while` позволяет выполнять оператор или блок до тех пор, пока условие не перестает быть истинным. Синтаксис его следующий:

```
while(выражение)
```

оператор

```
while(выражение)
```

```
{
```

блок операторов

```
}
```

Оператор или тело блока, связанного с `while`, не будет выполняться, если выражение изначально ложно.

Пример: Копирование строк, оканчивающихся нулем, с помощью оператора `while`.

```
void copy_string(char*dest,char*src)
{
    while(dest++=src++);
    //Здесь точка с запятой (;) является оператором
}
```

Оператор `for`.

Цикл `for` похож на цикл `while`, но дает вам две дополнительные возможности:

1.Вы можете включить в оператор инициализирующее выражение, исполняемое один раз перед тем, как будет произведена оценка условия.

2.Вы можете указать выражение, которое будет исполняться после каждой итерации оператора или блока, связанного с циклом `for`.

Синтаксис оператора следующий:

```
for(инициализация;    условное    выражение;    арифметическое
выражение)
```

оператор или блок операторов

Нужно иметь ввиду, что все три выражения являются необязательными, но знак ';' обязателен. Инициализирующее выражение, если есть, всегда будет выполняться; вычисление конечного выражения может не производиться, если условие ложно с самого начала.

Оператор `do/while`.

В цикле do/while оценка условия производиться после исполнения тела цикла. Это означает, что оператор или блок операторов, связанных с циклом, будет обязательно исполняться хотя бы один раз.

Синтаксис оператора имеет вид:

```
do
    оператор или блок операторов
    while(выражение);
```

3. Объект исследования, оборудование

Объектом исследования являются элементы среды программирования алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. Варианты заданий

Создание простейшей формы для программы решения квадратного уравнения.

Создание формы с выпадающим списком и выбором из него.

Создание формы с работающим компонентом TStringGrid.

Создание формы с работающим компонентом для размещения изображения.

Создание формы с работающим компонентом для редактирования текста, чтения из файла и его сохранения.

Создайте форму с кнопкой и реализуйте в качестве реакции функцию, в которой выполните следующее:

- объявить переменные и присвоить им значения в виде действительных констант соблюдая правила C++: 256; 2.56; -43000; 10^{12} ; 0.000000492; -10; -10^{-16} ; 16; 4.59016; -10000; 10^{17} ; 0.00006; -1; -10

- напишите выражения, соответствующие каждой из математических формул:

$$\text{а) } x+y^3; \quad \text{б) } (x+y)^{3,5}; \quad \text{в) } x^{4,3}; \quad \text{г) } \frac{a+b}{cd}.$$

– определите ошибки и запишите правильно следующие выражения:

$$\text{а) } \frac{x+2}{Y+4} \quad X+2.0/Y+4.0;$$

$$\text{б) } \frac{A*B}{C+2} \quad AB/'(C+2);$$

$$\text{в) } (A+B)(C+D) \quad A+B*C+D;$$

$$\text{г) } \frac{A}{B} + \frac{CD}{FGH} \quad A/B+CD/FGH;$$

5. Порядок выполнения работы

- Изучение теоретических вопросов.
- Выполнение задания преподавателя.
- Проверка выполненного задания.
- Анализ допущенных ошибок.
- Ответы на вопросы.
- Проверка отчета преподавателем.

6.Указания по оформлению отчета

Каждый студент составляет отчет, который должен содержать:

- Основные этапы решения задач на ЭВМ.
- Типы чисел, используемых в C++, форма их представления.
- Правила составления имен переменных на языке C++.
- Правила формирования арифметических выражений.
- Программирование задания, определенного преподавателем.

7.Контрольные вопросы

- Из чего состоит палитра компонентов?
- Объясните назначение инспектора объектов?
- Как в программе создается реакция на нажатие кнопки?
- Как определяется координата курсора мыши?
- Добавьте к существующему проекту новый текстовый модуль.
- Из чего состоит программа на языке C++?
- Есть ли пределы величины чисел, используемых при вычислении на ЭВМ?
 - Как записать возвведение числа в степень?
 - как можно ускорить процесс вычисления на C++?
 - Какие элементарные математические функции можно использовать в C++?

ЛАБОРАТОРНАЯ РАБОТА № 2

ХАРАКТЕРНЫЕ ПРИЕМЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ ПРИ РЕШЕНИИ ИНЖЕНЕРНЫХ ЗАДАЧ

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является:

- изучение основных этапов решения задач на ЭВМ;
- практика в разработке алгоритмов последовательной, разветвляющейся и циклической структуры;
- практика в использовании типовых приемов программирования;
- получение навыков отладки программ;
- углубление навыков практической работы на ЭВМ.

2. ОСНОВЫ ТЕОРИИ

2.1. Основные этапы решения задач на ЭВМ

Понятие “решение задачи” с помощью ЭВМ включает в себя гораздо больше, нежели просто вычисления на ЭВМ. Будет полезно представить себе основные этапы решения типичной инженерной задачи на ЭВМ и установить, что делает человек и что делает машина.

Постановка задачи и определение конечных целей

Этот вопрос касается выбора общего подхода, определения совокупности критериев, которым должна удовлетворять система, и задания условий ее работы.

В некоторых случаях это просто сделать, в других случаях на этот этап могут уйти месяцы работы. На данном этапе требуется глубокое понимание существа задачи. До настоящего времени использование ЭВМ на этапе постановки задачи было весьма ограниченным. Однако в последнее время наблюдается все более широкое использование ВТ, заключающееся в диалоговом режиме работы, при котором специалист непрерывно связан с машиной, может оперативно вмешиваться в ее работу, задавать вопросы и получать ответы на них в привычной ему терминологии.

Во многих случаях это облегчает и ускоряет правильную постановку задачи.

Математическое описание

Как правило, существует несколько способов математического описания задачи; должен быть выбран один из них или разработан новый способ, если ни один из имеющихся не приложим. Этот этап также требует полного понимания проблемы и знания соответствующих областей математики.

Численный анализ

Математическая формулировка задачи может оказаться непереводимой непосредственно на язык ЭВМ, так как она (ЭВМ) выполняет только арифметические действия и принимает простейшие логические решения. Такие обще известные математические понятия, как тригонометрические функции, дифференциальные уравнения, интегралы, квадратные корни, логарифмы - все это должно быть, выражено через элементарные арифметические операции. Более того, необходимо убедиться, что никакие погрешности, содержащиеся в исходных данных или внесенные в процессе вычислений, не влияют на точность результатов. Это целая область современной математики.

Программирование для ЭВМ

Численные алгоритмы решения задачи необходимо выразить в виде точно определенной последовательности операций ЭВМ. Обычно эта работа производится в две стадии. На первой стадии последовательность операций изображается графически в виде блок-схемы. Затем алгоритм нужно изложить на языке, который может быть “понят” ЭВМ непосредственно или после предварительного “перевода”. Такими языками являются, например, C++, Pascal, Java. Другими словами, обычно рисуют блок-схему метода решения; эта схема важна тем, что она дает ясную картину предстоящих операций, но не может быть “понята” машиной. Затем, используя блок-схему в качестве руководства, пишут программу на C++, которая будет “понята” машиной после некоторого предварительного перевода.

Отладка программы

В процессе программирования имеется столько возможностей допустить ошибку, что большинство программ работает сначала неверно. Ошибки в программе должны быть обнаружены и исправлены, а саму программу необходимо тщательно испытать, быть уверенным в достоверности результатов, на этом этапе используется ЭВМ.

Вычисления

На данном этапе можно производить расчеты по программе, используя исходные данные задачи. Как правило, расчет делается сразу для нескольких вариантов набора исходных данных, этот этап может занять от нескольких долей секунд до многих часов, в зависимости от задачи и от возможности ЭВМ.

Интерпретация результатов

Как уже говорилось, результаты вычисления, выдаваемые машиной, не всегда содержат полный “ответ” к задаче. Человек, производящий расчет на ЭВМ, должен каким-то образом интерпретировать результаты, чтобы понять, что они означают с точки зрения критериев, которым должна удовлетворять исследуемая система. Очень часто бывает нужно частично или полностью повторить предшествующие этапы, пока задача не будет действительно решена.

Из этого краткого рассмотрения можно сделать некоторые выводы. Во-первых, ЭВМ сама задач не решает, она только производит заранее заданные последовательности вычислений. Во-вторых, использование ЭВМ заставляет уделять гораздо большее внимание осмысливанию решаемой задачи. Машина может производить вычисления быстрее и точнее человека, но она не способна решать, какова должна быть программа вычислений или что делать с получаемыми результатами. В-третьих, ЭВМ никоим образом не снимает необходимости детального изучения исследуемой области и применяемых для решения задачи разделов математики.

2.2. Запоминание результатов

В рассмотренных ранее задачах результатом вычисления было значение простой переменной, для записи которой в памяти ЭВМ выделяется одна ячейка. Если в процессе вычислений значение временной изменяется, то в памяти ЭВМ после окончания вычислений остается лишь последнее значение результата. Чтобы запомнить все промежуточные значения, нужно выделить для их решения необходимое количество ячеек памяти (массив), а текущий результат обозначить переменной с индексом.

2.3. Вычисление суммы и произведения

Если необходимо вычислить сумму значений некоторой функции $y=f(x)$ при различных значениях аргумента, целесообразно организовать цикл, в котором надо предусмотреть не только вычисление значений функции, но и накопление сумм путем прибавления полученных слагаемых к сумме всех предыдущих слагаемых. Формула, используемая для накопления суммы, имеет вид:

$$S_n = S_{n-1} + y_n$$

Поскольку надобности в запоминании значений всех слагаемых и промежуточных сумм нет, в качестве S и y нужно использовать простые переменные и накопление суммы вести в цикле по формуле

$$S = S + y,$$

где знак " $=$ " означает присваивание значения. Если начальное значение S предварительно приравнять нулю, то после первого выполнения цикла значение S будет равно первому значению функции.

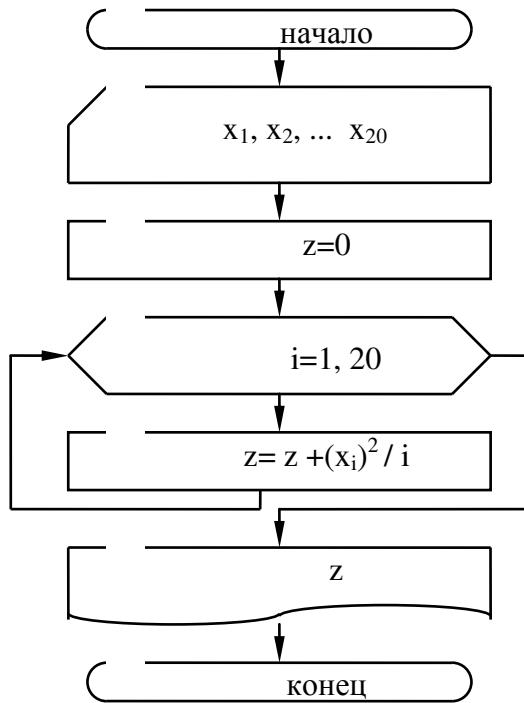
Аналогично накапливается и произведение (например при вычислении факториала) с той лишь разницей, что для его накопления используется формула

$$P = P \cdot Y,$$

а начальное значение произведения должно быть равно единице.

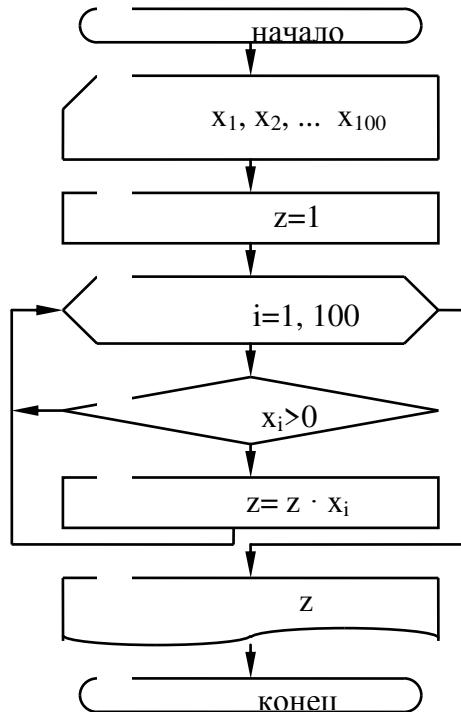
Например, вычислить значение функции $Z = \sum_{i=1}^{20} \frac{x_i^2}{i}$, где x_i - элемент массива.

Схема алгоритма решения этой задачи имеет вид:



Блок 3, задающий начальное значение суммы, стоит перед циклом, в котором накапливается сумма. Блок 5 вычисляет значение слагаемого и накапливает эту сумму. Поскольку результат решения этой задачи одно число, блок печати стоит за циклом и выполняется один раз.

Рассмотрим еще один пример: вычислить произведение положительных элементов массива (x_1, x_2, \dots, x_{100}).



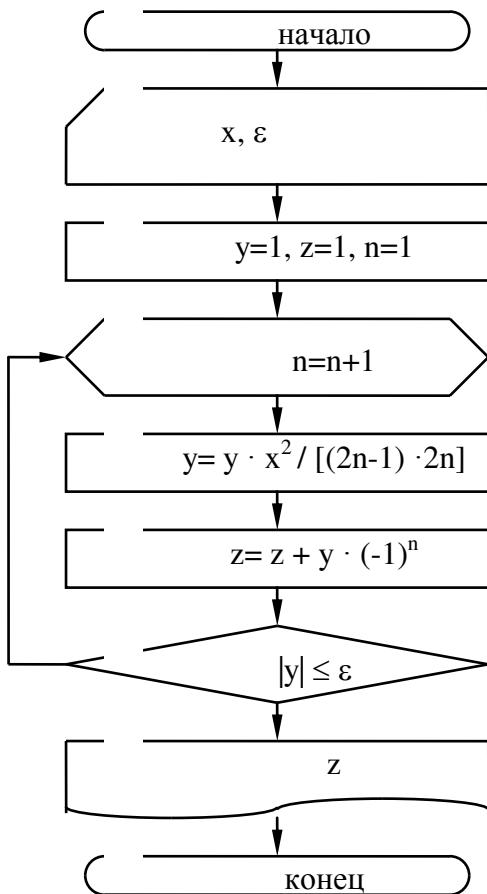
В данном случае значения сомножителей вычислять не требуется, поскольку они уже имеются в массиве. Однако, прежде чем накапливать произведение, надо проверить, является ли сомножитель положительным (блок 5). Блок 3 задает начальное значение произведения, равное 1. В одной из ветвей этого процесса стоит блок 6, осу-

ществляющий накопление произведения. При невыполнении условия $x_i > 0$ никаких действий не предусматривается, а осуществляется переход к началу цикла.

2.4. Вычисление суммы членов бесконечного ряда

Задачи этого вида являются типичными задачами, использующими итерационный цикл, т.к. заранее не известно, при каком члене ряда будет достигнута требуемая точность. Выход из цикла организуется по условию достижения требуемой точности. Для вычисления суммы членов ряда используется рассмотренный ранее прием накопления суммы.

Схема алгоритма решения этой задачи представлена ниже:



Блок 3 задает начальное значение y , равное 1, начальное значение суммы, равное члену ряда с номером 0 и начальное значение параметра цикла. В блоке 5 вычисляется значение текущего члена ряда. Блок 6 накапливает сумму. Блок 4 изменяет параметр цикла. Блок 7 проверяет условие повторения цикла и осуществляет переход к началу цикла, если $|y| > \epsilon$, или выход из него в противном случае.

2.5. Нахождение наибольшего и наименьшего ряда чисел

Нахождение наибольшего (наименьшего) значения в заданном или вычисляемом ряде чисел выполняется в цикле, в котором каждое число из данного ряда сравнивается с наибольшим (наименьшим) из всех предыдущих. Если очередное число окажется больше наибольшего из предыдущих значений, то его надо считать новым наибольшим значением, в противном случае наибольшее число останется прежним. Сказанное можно описать условной математической формулой:

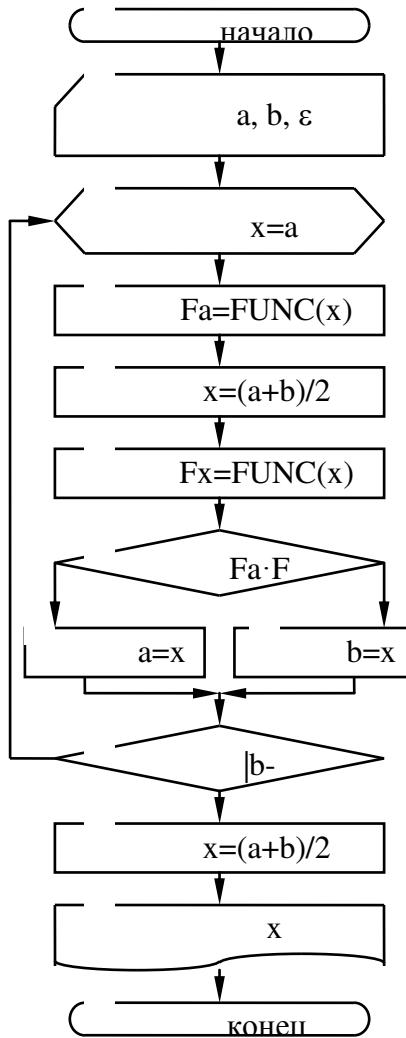
$$y_{\max} = \begin{cases} y_i, & \text{если } y_i > y_{\max} (i = 1, 2, \dots, n); \\ y_{\max}, & \text{если } y_i \leq y_{\max} (i = 1, 2, \dots, n). \end{cases}$$

Аналогично для наименьшего значения:

$$y_{\min} = \begin{cases} y_i, & \text{если } y_i < y_{\min} (i = 1, 2, \dots, n); \\ y_{\min}, & \text{если } y_i \geq y_{\min} (i = 1, 2, \dots, n). \end{cases}$$

В качестве начального значения y_{\max} и y_{\min} целесообразно брать какое-либо известное значение Y , например Y_1 .

Алгоритм нахождения наибольшего значения в заданном массиве 10 чисел имеют следующий вид:



Следует отметить, что здесь речь идет не о нахождении максимума или минимума функций с заданной переменной, а о наибольшем или наименьшем значении из заранее заданных или вычисленных в данной программе чисел. В связи с этим при нахождении наибольшего (наименьшего) значения из вычисленных значений функции по данному алгоритму истинный максимум (минимум) функции может находиться между исследуемыми дискретными значениями. Очевидно, что для нахождения экстремума функции необходимо строить итерационный процесс.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы алгоритмизации и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

Каждый вариант содержит три задачи. В задаче 1 во всех вариантах необходимо вычислить значение арифметических выражений при заданных значениях переменных. Задача 2 состоит из трех отдельных задач, в которых должны быть вычислены значения условного арифметического выражения. В задаче 2а выражение вычисляется при одном значении аргумента. В задачах 2б и 2в необходимо найти (протабулировать) функцию соответственно при произвольном заданном наборе аргументов и на отрезке при изменении аргумента с заданным постоянным шагом. При решении задачи 2б использовать цикл с заданным числом повторений. При решении задачи 2в - итерационный цикл. Вывод результатов решения задач 2б и 2в оформить в виде таблицы с тремя столбцами и с заголовком в следующем виде:

Номер строки	Аргумент	Функция

Задача 3 формулируется для каждого варианта. Разработанную программу проверить решением тестовой задачи ,исходные данные для которой задать самостоятельно.

Вариант 1

Задача 1.1

$$S = \frac{\sin^3 3x + \operatorname{arctg} \varphi^3 - 2 \cdot 10^{3,1}}{\sqrt{|a^3 + r^3|} + \sqrt[5]{|a^3 + r^3| x^2}} + \ln \left| \operatorname{tg} \left(x - \frac{\pi}{2} \right) \right|;$$

$$\psi = \cos \frac{\pi(x+1,3)}{\sqrt{\delta^2 + 18x^2}} + x^\varphi - \lg |\delta^2 + 18x^2| + S^2,$$

где $x=1,184$; $\varphi=4,3$; $a=12$; $r=2,3 \cdot 10^4$; $\pi=3,14159$; $\delta=0,5$.

Задача 1.2

$$Z(\alpha) = \begin{cases} \lg \sqrt{\frac{\cos^3(\pi-\alpha)}{\varphi^2 + \psi^4}} & \text{при } \alpha > \varphi \\ \frac{\varphi}{1,3} \operatorname{ctg}^3 \left(\alpha - \frac{1}{\psi} \right) & \text{при } \alpha < \varphi \end{cases}$$

где $\pi=3,14$; $\varphi=9,3$; $\psi=10$.

- a) $\alpha=12,3$;
- б) $\alpha=(-1; 7; 12; 19)$,
- в) $\alpha \in [a, b]$, $a=1$; $b=3,4$, $\Delta \alpha=0,3$.

Задача 1.3

Построить матрицу $A_{m \times n}$, элементы которой вычисляются по формуле

$$a_{ij} = \begin{cases} \ln(i/2 + \sqrt{j}), & \text{если } (i+j) - \text{четное} \\ \sin \frac{i+j}{\sqrt[3]{i}}, & \text{если } (i+j) - \text{нечетное} \end{cases}$$

Вариант 2

Задача 2.1

$$y = \frac{\cos x^{1/3}}{\sin^3(a + x^{1/3})} + \ln \left| \frac{\sqrt{|\operatorname{tg}^3 \varphi| + 1,3}}{\sqrt{|x^{1/3} - 2a|}} \right|$$

где

$$\varphi = \left(e^{bx} - e^{-zt} \right)^3 + \arcsin \frac{3^x + \sin^3(\pi - t^2)}{\varphi^4 + e^6},$$

$x=4,3$; $a=10$; $b=2,1 \cdot 10^{-3,1}$; $z=-414 \cdot 10^{-2}$; $t=0,5$; $\psi=12$; $\pi=3,14159$.

Задача 2.2

$$y(t) = \begin{cases} \arctg^{\frac{3}{4}} \left(\frac{\cos \frac{axt}{7} + \sqrt{\psi}}{t^3 + \sqrt[4]{|t+b|}} \right), & \text{при } t < x \\ \lg \left(e^{-\pi t} + ax^{\frac{1}{3}} e \right), & \text{при } t \geq x \end{cases}$$

где

$$a=2,31; \quad x=11,3; \quad b=12; \quad \pi=3,1416; \quad \varphi=21 \cdot 10^{-2}.$$

$$t=2,91;$$

$$t=(1; 2; 7; 19; 24.)$$

$$t \in [t_1, t_2], \quad t_1=2, \quad t_2=14, \quad \Delta t=2.$$

Задача 2.3

Вычислить матрицу , равную сумме двух матриц $A_{m \times n}$ и $B_{m \times n}$.

Вариант 3

Задача 3.1

$$\alpha = \arcsin^3 \frac{x-1,3}{10^5 \sqrt{|x-1,3|}} + \frac{e^{\frac{|x-1,3|}{a}} + \cos \ln |b^3|}{4,3 \cdot 10^{1,8} \operatorname{ctg} \left(\pi - \frac{1}{x^3} \right)},$$

$$V = \lg 4 \sqrt{\frac{\cos^6 \alpha}{1+\alpha^2}} - \frac{z}{2} \ln |\cos^5 \alpha^3|;$$

$$\text{где } x=4,18; \quad \pi=3,14159; \quad a=180; \quad b=-2,1 \cdot 10^3; \quad z=4 \cdot 10^{-3}.$$

Задача 3.2

$$V(x) = \begin{cases} \arcsin^3 \frac{x-a\varphi}{\sqrt{|x+2,1| \cdot 10^4}}, & \text{при } x > a \\ \frac{b}{2} \ln |\cos^5(\pi-x)|, & \text{при } x \leq a \end{cases}$$

$$\text{где } a=-6; \quad \varphi=4,45; \quad b=21 \cdot 10^{-3}; \quad \pi=3,14;$$

$$\text{a) } x=7.$$

$$\text{b) } x=(-8, -7, 4, 7, 17).$$

$$\text{c) } x \in [x_1, x_2], \quad x_1=-6, \quad x_2=9, \quad \Delta x=3.$$

Задача 3.3

Вывести на печать положительные элементы главной диагонали матрицы $X_{n \times n}$ и их сумму .

Вариант 4

Задача 4.1

$$V = \cos^3\left(2\varphi + \frac{\pi}{4}\right) + \frac{\operatorname{tg}^5 |\ln x^3| + 4,2 \cdot 10^{-2,8}}{\sqrt{e^{\sin(2\varphi + \frac{\pi}{4})} + \sqrt{|\psi + e|}}};$$

$$Z = \sqrt[3]{\left| \operatorname{arctg}^3 V + \left| \psi^3 \right| \right|}$$

где $\varphi=2,14$; $\pi=3,14159$; $x=10$; $\psi=0,5$.

Задача 4.2

$$Z(x) = \begin{cases} 2^{\frac{a}{x}} + \sqrt[3]{\sin^2 5x + \varphi}, & \text{при } x < \varphi \\ \cos \frac{\pi(x+b)}{\sqrt[7]{x^2 + a^2}}, & \text{при } x \geq \varphi \end{cases}$$

где $a=4,2$; $\varphi=11$; $\pi=3,14159$; $b=18,3 \cdot 10^2$,a) $x=7,3$.b) $x=(1, 4, 7, 15, 21)$.c) $x \in [x_1, x_2]$, $x_1=-2$, $x_2=8$, $\Delta x=2$.

Задача 4.3

Вывести на печать номера элементов массива (Y_1, Y_2, \dots, Y_{100}), удовлетворяющих условию $0 < Y_i < 1$, и их сумму.**Вариант 5**

Задача 5.1

$$r = e^{|\sin x^3|} \ln(1 + \cos^2 x) + \frac{\operatorname{tg}^3(1 + \cos^2 x) + 4,8 \cdot 10^{1,3}}{\sqrt{a^2 + b^2} + \sqrt[5]{1 + \cos^2 x}};$$

$$\alpha = \operatorname{arctg}^{\frac{3}{5}} \frac{\lg |\cos(\pi - r)|}{e^{\sqrt{|x|}} + a^{b^2}}$$

где $x=-4,3$; $a=13$; $b=2,4 \cdot 10^{-3}$; $\pi=3,14$.

Задача 5.2

$$V(s) = \begin{cases} s^\varphi \ln |\cos^3(\pi - sb)| \\ \arccos \frac{s - \sqrt{|a-s|}}{2 + a^s \cdot 10^4} \end{cases} \text{ при } s \leq \varphi;$$

где $\pi=3,14159$; $\varphi=41$; $b=21 \cdot 10^{-1}$; $a=8,121$;a) $s=40$;б) $s=(-6,4; 14; 46)$;в) $s \in [s_1; s_2]$; $s_1=1$; $s_2=18$; $\Delta s=4$.

Задача 5.3

Вывести на печать минимальный элемент массива ($x_1; x_2; \dots; x_{10}$) и его номер .

Вариант 6**Задача 6.1**

$$z = \operatorname{tg} \frac{e^{t^2} + 3^x}{\sqrt{|t^2 + a|}} - \frac{\cos^3 \ln|\psi^3| + 2,8 \cdot 10^{-3,4}}{\sqrt{\sin^3 \left(\pi - \frac{\varphi}{2} \right)} + \sqrt{|t^2 + a|}};$$

где $a=4,8 \cdot 10^{-2}$; $\varphi=-0,51$; $\pi=3,14159$

Задача 6.2

$$d(r) = \begin{cases} \sqrt[5]{|\cos^3 r^2 + a\varphi^2|}, & \text{при } r < \varphi \\ \lg|r^3 - e^\pi|, & \text{при } r > \varphi \end{cases}$$

где $\pi=3,14159$; $a=14$; $\varphi=2,7 \cdot 10^{-3}$

- а) $r=2,8$
- б) $r=(-4; 3; 4,1; 7; 9,2; 13)$
- в) $r \in [r_1, r_2]$, $r_1=-2$, $r_2=5$, $\Delta r=0,5$.

Задача 6.3

Вычислить, запомнить и напечатать положительные значения функции $Y=n \cdot \sin(x) - \cos(nx)$ для X , изменяющегося от X_1 до X_2 с шагом ΔX .

Вариант 7**Задача 7.1**

$$k = 4x \sqrt[4]{|\varphi - a^2|} + \frac{\ln^3 |\sin b^2| + \sqrt{\operatorname{tg}^2 |\varphi - a^2| + 1,3}}{2,3 \sin^{\frac{3}{2}} |\ln|z||}$$

$$y = \ln \frac{e^k - ne^\pi}{\sqrt[3]{|k|} + 18,1}$$

где $x=8,12$; $\varphi=10$; $a=7,5 \cdot 10^{-3}$; $b=3,8 \cdot 10^{4,8}$;
 $z=-0,141$; $n=14$; $\pi=3,14159$.

Задача 7.2

$$\delta(x) = \begin{cases} \ln \left| \operatorname{tg} \left(x - \frac{\pi a}{2} \right) \right|, & \text{при } x < a \\ \cos \frac{\gamma(x+2,4)}{\sqrt[3]{b^2 + x^2}}, & \text{при } x \geq a \end{cases}$$

где

$$\pi=3,14159; \quad \gamma=12; \quad b=121 \cdot 10^{-4}; \quad a=4,46,$$

а) $x=2,51$

б) $x=(1,2; 4,5; 8,9; 17,1)$

в) $x \in [x_1; x_2]; x_1=-5; x_2=6; \Delta x=1.$

Задача 7.3

Вычислить сумму 20 первых элементов ряда $S=1+x+x^2/2+\dots+x^{19}/19$, используя для определения x^i прием накопления произведения, т.е. $x^i=x^{i-1} \cdot x$.

Вариант 8

Задача 8.1

$$f = \sqrt{x} e^{\sqrt[3]{|t|}} + \frac{\ln(e^{-t} + |a|e^{t+2}) + 4,1 \cdot 10^{2,1}}{b^{\sin \psi} + \ln \left| \operatorname{tg} \left(\frac{\pi}{4} + \frac{x}{2} \right) \right|}$$

где

$$a = \arccos^{\frac{2}{3}} \frac{\sin \left(\frac{\pi}{4} + \frac{x}{2} \right)}{t^3 + \sqrt[3]{|t+1|}};$$

$$x=2,11; \quad t=14 \cdot 10^{-2,3}; \quad \psi=0,64; \quad b=19; \quad \pi=3,14.$$

Задача 8.2

$$t(x) = \begin{cases} \operatorname{tg} e^\pi + \sqrt{z^2 + x}, & \text{при } x > z \\ \operatorname{arctg} x^3 + \frac{a+b}{\cos z^2}, & \text{при } x \leq z \end{cases}$$

где $z=2,41; \quad a=2,1 \cdot 10^{-9}; \quad b=14; \quad \pi=3,14159$.

а) $x=1,3;$

б) $x=(-3; 1; 2; 7,1; 8,3.)$;

в) $x \in [x_1; x_2]; x_1=-4; x_2=5; \Delta x=1.$

Задача 8.3

Вычислить среднее арифметическое отрицательных элементов двумерной матрицы A_{mn} , полагая, что в ней имеются отрицательные значения, и вывести его на печать .

Вариант 9

Задача 9.1

$$y = \frac{2^x + 4,1 \cdot 10^{-1,7}}{(x - \pi) \sqrt{\sin^2 5x + 1,2}} + \frac{\operatorname{tg}^3(\varphi) - \lg \sqrt{|a^2 + r|}}{c \operatorname{tg} |a^2 + r|^{\frac{3}{2}} + e^{|r|}}$$

где

$$r = \cos \frac{\pi(x + 4,2)}{\sqrt[7]{|a^2 + \pi|}}; \\ x = -4,87; \pi = 3,14; \varphi = -2,9; a = 10.$$

Задача 9.2

$$M(t) = \begin{cases} at^2 - \cos^3(\pi - t^2)^2 & \text{при } t < a; \\ 2,3t\cos^3 \beta t & \text{при } t \geq a. \end{cases}$$

где $\pi = 3,14$; $a = 12$; $\beta = 2,41 \cdot 10^{-3}$.

- а) $t = 4,31$;
 б) $t = (-4; 1; 2; 8,6; 23; 41,6)$;
 в) $t \in [t_1; t_2]$; $t_1 = -6$; $t_2 = 10$; $\Delta t = 2$.

Задача 9.3

В каждом столбце двумерной матрицы A_{mn} , найти наименьший по модулю элемент и его координаты и вывести их на печать.

Вариант 10

Задача 10.1

$$\psi = e^{i g^2 x} + \frac{\sqrt[3]{|t - z^2|}}{\cos^3 \pi^2 + e^{\varphi^2}},$$

$$\varphi = \frac{\sqrt{|t + z^2|} + 2,3 \cdot 10^{1,6}}{\ln |t^3 + 1,3|}; \quad \pi = 3,14159; t = 2 \cdot 10^{-3}; x = 14; z = -1,24.$$

Задача 10.2

$$y(x) = \begin{cases} \frac{1}{a} (x + b e^{px}) & \text{при } x > b, \\ \sqrt[3]{x^2 + |\cos^3(\pi b)|} & \text{при } x \leq b \end{cases}$$

где

$$b = 4,18; a = 12; p = 2 \cdot 10^{-4}; \pi = 3,14159.$$

- а) $x = 2,1$;
 б) $x = (-1; 3,1; 8; 10; 17,1)$.

в) $x \in [x_1; x_2]$; $x_1=-3$; $x_2=7$; $\Delta x=0,5$.

Задача 10.3

Найти наименьшее значение функции $y=a \cdot e^{-bx} \cdot \sin(\omega x+\varphi)$ в интервале изменения аргумента X от 0 до С с шагом X и вывести его на печать .

Вариант 11

Задача 11.1

$$Z = \frac{5,2 \cdot 10^{3,2} + \lg(e^{-t} + |a|e^{t+2})}{\lg \left| \operatorname{tg} \left(\frac{\pi}{2} + \frac{x}{4} \right) \right|}$$

где $x=2,3$, $\varphi=0,64$, $b=19$, $\pi=3,14159$, $t=2,41$,

$$a = \arccos \frac{\frac{2}{3} \sin \left(\frac{\pi}{2} + \frac{x}{4} \right) \cdot 10^{-4}}{\sqrt[3]{|t+1|} + t^3}.$$

Задача 11.2

$$F(x) = \begin{cases} ax^2 - \cos^3(\pi - x)^2 & \text{при } x < a \\ 2,1x \cos^3 \beta x & \text{при } x \geq a \end{cases},$$

где $\pi=3,14$, $a=10$, $\beta=2,31 \cdot 10^{-3}$

- а) $x=4,23$,
- б) $x = (4,9, 2,8, 21, 41,3)$
- в) $x \in [x_1, x_2]$, $x_1=-6$, $x_2=11$, $\Delta x=2$.

Задача 11.3

Вычислить и вывести на печать среднее геометрическое положительных элементов одномерного массива. Если таких чисел нет, то вывести текст "ПОЛОЖИТЕЛЬНЫХ ЭЛЕМЕНТОВ НЕТ "

Вариант 12

Задача 12.1

$$Y = \frac{2,9 \cdot 10^{-3,1} + \cos^3 \ln |\psi^3|}{\sqrt[5]{\sin^2 \left(\pi - \frac{\varphi}{2} \right)} + \sqrt[4]{|t^2 + a|}} - \operatorname{tg} \frac{e^{t^2} + 3^x}{\sqrt{|t^2 + a|}},$$

$$\psi = \operatorname{arctg} \frac{\sin^{-1} 2x + ctgt^3}{\frac{1}{t^3} + 8,74}, t = 2,14, x = 10, a = 4,8 \cdot 10^{-2}$$

где $\varphi=0,51$, $\pi=3,14$.

Задача 12.2

$$Z(t) = \begin{cases} \sqrt[5]{\cos^3(\pi b) + t^2} & \text{при } t \leq b \\ \frac{1}{a}(t + be^{pt}) & \text{при } t > b \end{cases}$$

где $b=4,2$, $a=11$, $p=2 \cdot 10^{-4}$, $\pi=3,14$.

- a) $t=2,1$;
- б) $t=(-1; 3,2; 8,1; 10; 17,8)$.
- в) $t \in [t_1; t_2]$; $t_1=-3$; $t_2=7$; $\Delta t=0,5$.

Задача 12.3

Вычислить сумму членов ряда

$$S = x^n + x^{(n-1)}/2 + x^{(n-2)}/3 + \dots + x/n+1/(n+1) + (x-1)/(n+2) + \dots$$

с точностью до члена ряда , меньшего 10^{-6} . Для определения текущего члена ряда использовать рекуррентную формулу.

5.ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

1. Изучить теоретические вопросы.
2. Решить заданные варианты задач:
 - составить алгоритм вычисления;
 - составить программу расчета;
 - отладить программу;
 - провести вычисления по составленной программе.
3. Проанализировать результаты расчетов.
4. Ответить на контрольные вопросы.

6 УКАЗАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Каждый студент оформляет отчет который должен содержать:

- постановку задачи;
- алгоритм решения задачи;
- распечатки отладок;
- распечатку результатов расчета в наиболее наглядной форме;
- анализ результатов;
- перечень и характеристика ошибок ,допущенных в процессе прохождения задания;
- ответы на контрольные вопросы.

При составлении программы следует дать по необходимости пояснение использованных имен переменных .

7.КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные этапы подготовки к решению задач на ЭВМ.
2. Что такое алгоритм решения задачи?

3. Какие типы циклических вычислительных процессов вы знаете?
4. Можно ли использовать оператор цикла в итерационном цикле?
Объясните.
5. Что такое отладка программы? Перечислите типы ошибок, возникающих при программировании.

ЛАБОРАТОРНАЯ РАБОТА № 3-4

ВВОД И ВЫВОД ИНФОРМАЦИИ

1. Цель и задачи работы

При выполнении работы студенты знакомятся с операторами ввода и вывода информации. Изучают способы ввода и вывода простых переменных, массивов, текстовой информации.

Студентам необходимо самостоятельно написать на С++ ряд программ, обеспечивающий ввод и вывод информации.

2. Основы теории

2.1. Консольный ввод и вывод.

Язык С++ не определяет ввод-вывод, но предоставляет широкий набор средств для реализации ввода-вывода. Можно использовать как средства, обеспечиваемые конкретной реализацией библиотеки `iostream`, так и разрабатывать собственные средства. В С++ файл рассматривается как последовательность байтов, а набор функций библиотеки обеспечивает возможность обработки данных различных размеров и форматов. Ввод-вывод в С++ - это широкое использование перегрузки. Используя перегрузку операций `<<` и `>>`, библиотека `iostream` дает возможность ввода и вывода данных. Можно выводить данные как встроенных типов, так и типов, определенных пользователем. Программист может использовать и новые средства ввода-вывода языка С++, и старые, те, что известны в традиционном Си. Более предпочтительно использовать новые средства, предоставляемые библиотекой `iostream`. Этот ввод-вывод проще и элегантнее.

Для организации ввода-вывода в программе на С++ используется файл включения `iostream.h`, который содержит определение нескольких классов и объектов, обеспечивающих методы ввода-вывода. В частности, язык поддерживает два предопределенных объекта, выполняющих ввод-вывод:

`cout` - объект класса `ostream`. Этот класс используется для управления форматизированным и неформатизированным выводом. Здесь определяется операция `<<` для вывода;

`cin` - объект класса `istream`. Этот класс используется для управления форматизированным и неформатизированным вводом. Здесь определяется операция `>>` для ввода.

Таким образом, для ввода достаточно в программе написать выражение:

`cin >> идентификатор;`

По этому выражению из входного потока читается последовательность символов до пробельного символа, преобразуется к типу “идентификатор” и помещается в “идентификатор”.

```
char text[20]; // объявление массива из 20-ти символов
int name; // объявление простой целой переменной
cin >> text;
cin >> name;
```

Для вывода информации в выходной поток используется выражение вида:

```
cout << значение;
```

“Значение” преобразуется в последовательность символов и выводится в выходной поток.

Так как >> и << - это знаки операции, то “cin>> переменная” и “cout<< значение”- это выражения. Значением выражения со знаком операции ввода является ссылка на объект cin, а значением выражения со знаком операции вывода является ссылка на объект cout. Из этого следует, что несколько выражений с операцией вывода можно объединить в одно выражение. Например,

```
cout << "a+b=" << a+b << '\n';
```

Объединенный оператор вывода может быть, расположен на нескольких строках:

```
cout << "сумма a+b ="
    << a+b
    << '\n';
```

Выражения со знаками операции ввода также могут быть объединены в одно:

```
cin >> val1 >> val2;
```

При наборе данных на клавиатуре значения для такого оператора должны быть разделены пробельными символами (пробелы, символы табуляции, символы новой строки).

2.2. Ввод и вывод файлов.

Для того чтобы использовать потоковый ввод и вывод файлов надо подключить библиотеку #include < fstream.h>. В данной библиотеке имеется несколько конструкторов для создания объектов классов:

ofstream- класс для вывода в файл;

ifstream- класс для ввода из файла;

Первый конструктор каждого из этих классов создает файловую переменную, но не связывает ее с файлом.

```
ofstream f1;
```

Объект этого класса содержит в своем составе некоторые данные и некоторые функции.

В этих классах есть поле, которое должно содержать название файла, в который будет осуществляться вывод. Кроме того, есть поля,

содержащие способ открытия файла (текстовый или бинарный), по умолчанию файл открывается в текстовом режиме. Данный класс содержит различные функции. Например, имеется функция открытия файла:

```
f1.open ("Название файла");
f1.close ();
f1 << // переопределенный оператор вывода файла
```

Рассмотрим пример вывода файлов:

```
#include<fstream.h>
#include<iostream.h>
void main ()
{
    float x ;
    int y;
    ofstream f1;
    cout <<"x="; cin >>x;
    cout <<"y="; cin>y;
    f1.open ("exampl.dat");
    if (!f1) // проверка правильности открытия
    {
        cout<<"Ошибка открытия файла "; abort();
    }
    f1 <<"x="<<x <<"\n y="<<y;
    f1.close ();
}
```

Рассмотрим чтение файла. Предположим, что имеется файл, в котором записаны два числа. Необходимо прочитать из файла эти числа и присвоить их двум переменным float x и int y.

```
#include<fstream.h>
#include <iostream.h>
void main()
{
    float x;
    int y;
    ifstream f1; // для чтения
    f1.open ("exampl");
    if (f1==0)
    {
        cout<<"Ошибка открытия файла";
        abort ();
    }
    f1>>x >>y;
    f1. close ();
    cout <<"x="<<x <<" y="<<y; //для проверки правильности чтения
}
```

Классы ofstream и ifstream имеют несколько конструкторов. Второй конструктор не требует функций открытия и закрытия файлов. Он сразу в

момент создания объекта открывает файл, но для этого надо использовать конструктор в следующей форме:

```
ofstream f1("имя файла");
ifstream f2("имя файла");
```

Например:

```
ofstream f1("example");
```

Данный конструктор создает файловую переменную f1, открывает файл и связывает с ним переменную. Необходимо проверить правильность открытия файла, прежде чем осуществлять ввод или вывод. Использовать функцию закрытия файла в явном виде тоже необязательно, т.к. деструктор автоматически закрывает файл при окончании работы программы.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы алгоритмического языка С++: операторы ввода и вывода информации.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ.

1. Записать оператор вывода на печать заголовка “Исходные данные” и затем таблицу значений переменных с их наименованиями:

E=0,574	T=121,876
A=2,0	B=7,5
K=0.0987	L=87

2. Ввести и вывести на печать массив исходных данных из пяти целых элементов

K=[-1, 6, 18, 765, 73].

3. Создать файловую переменную f1 и связать ее с файлом.

4. Прочитать из какого-то файла два числа и присвоить их двум переменным int x и float y.

5. Прочитать из файла 2 дробных числа, сложить их и записать файл в файл результатов.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

- Изучение теоретических вопросов.
- Составление программ по заданию преподавателя.
- Проверка составленных программ.
- Анализ и исправление допущенных ошибок.
- Ответы на контрольные вопросы.
- Проверка отчета преподавателем.

6. УКАЗАНИЯ ПО СОСТАВЛЕНИЮ ОТЧЕТА.

- Каждый студент составляет отчет, который должен содержать:
 - Правила использования операторов ввода и вывода.
 - Правила ввода и вывода типов, определенных пользователем.
 - Правила ввода и вывода файлов.
 - Правила чтения из файла.
 - Программы решения задач, заданных преподавателем.
 - Ответы на контрольные вопросы.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ.

- Как ввести и вывести на экран числа?
- Как ввести тип данных, определенный пользователем?
- Как создать объекты классов?
- Как прочитать из файла число?
- Как это число присвоить переменной int y?

ЛАБОРАТОРНАЯ РАБОТА № 5

ПОИСК ЭКСТРЕМУМА ФУНКЦИЙ

1. Цель работы

Целью данной работы является:

- практика в разработке алгоритмов, содержащих
- итерационные циклы;
- использование методов минимизации функций при
- решении задач численного анализа;
- практика программирования;
- накопление опыта оформления результатов вычисления;
- углубление навыков отладки программ и практической работы на ЭВМ.

2. Основы теории

Рассмотрим вначале методы поиска экстремума функции одной переменной. Эта одномерная задача широко распространена в практике, кроме того, большинство методов решения многомерных задач сводится к поиску одномерного минимума.

Будем считать, что анализируемая функция задана на отрезке $a \dots b$, и имеет на этом отрезке (включая и его краевые точки) только один локальный минимум. Простейший итерационный процесс, сходящийся к этому минимуму, имеет следующую структуру. Вначале интервал, в котором ищется минимум функции $y = f(x)$, разбивается на ряд отрезков равной длины (рис. 4). Далее сравниваются значения функции на границах первого отрезка $f(x_1)$ и $f(x_2)$. Если $f(x_2) < f(x_1)$, то осуществляется переход ко второму отрезку. Если $f(x_3) < f(x_2)$, то переходят к третьему отрезку и т.д. Указанную процедуру повторяют до тех пор, пока не будет пройден минимум функций, т.е. условие не будет выполняться ($f(x_{i+1}) > f(x_i)$). После этого шаг аргумента уменьшается в N раз и изменяется его знак $\Delta X = -\Delta X / N$.

Дальнейшее изменение аргумента осуществляется в противоположном направлении с более мелким шагом, при этом условие перехода к следующему шагу остается прежним и т. д . до достижения заданной погрешности вычислений.

В практике обычно используется частный случай рассмотренного способа нахождения минимума функции – способ половинного деления шага, т.е. при $N=2$, обеспечивающий наименьшее число шагов поиска.

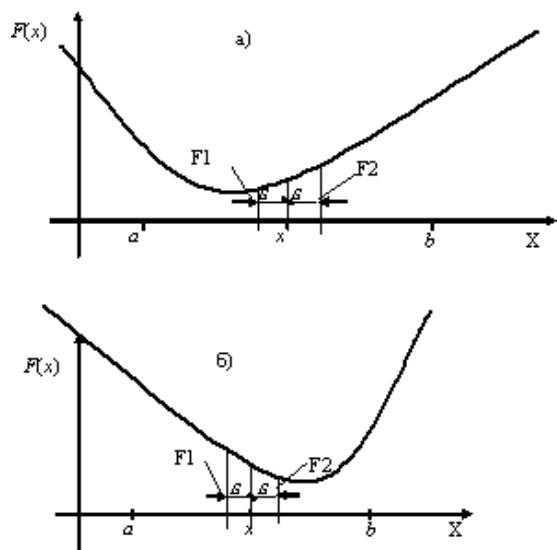


Рис.1. Численное нахождение минимума методом деления интервала поиска

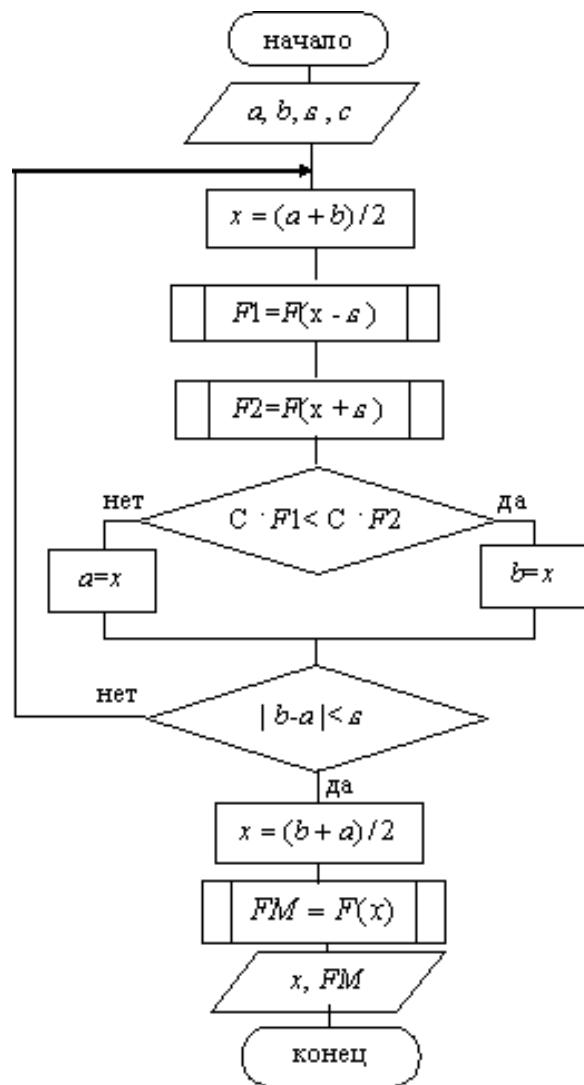


Рис. 2. Схема алгоритма метода половинного деления

Метод хорд

Метод основан на замене функции на каждом шаге поиска хордой, пересечение которой с осью дает приближение корня.

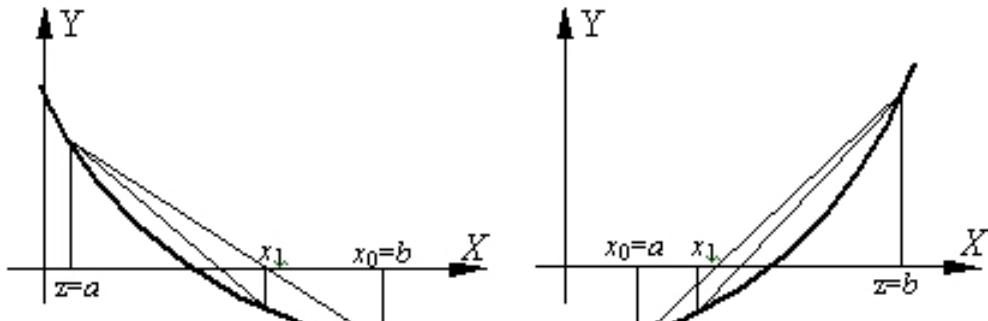


Рис. 3. Метод хорд

При этом в процессе поиска семейство хорд может строиться:

- при фиксированном левом конце хорд, т.е. $z=a$, тогда начальная точка $x_0=b$ (рис. 3а);
- при фиксированном правом конце хорд, т.е. $z=b$, тогда начальная точка $x_0=a$ (рис. 3б);

В результате итерационный процесс схождения к корню реализуется рекуррентной формулой:

для случая а):

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(a)}(x_n - a);$$

для случая б):

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(b)}(x_n - b);$$

Процесс поиска продолжается до тех пор, пока не выполнится условие $|x_{n+1} - x_n| \leq \varepsilon$ или $|h| \leq \varepsilon$.

Метод обеспечивает быструю сходимость, если $f(z) * f''(z) > 0$, т.е. хорды фиксируются в том конце интервала $[a,b]$, где знаки функции $f(z)$ и ее кривизны $f''(z)$ совпадают.

Схема алгоритма уточнения корня методом хорд представлена на рис.4.

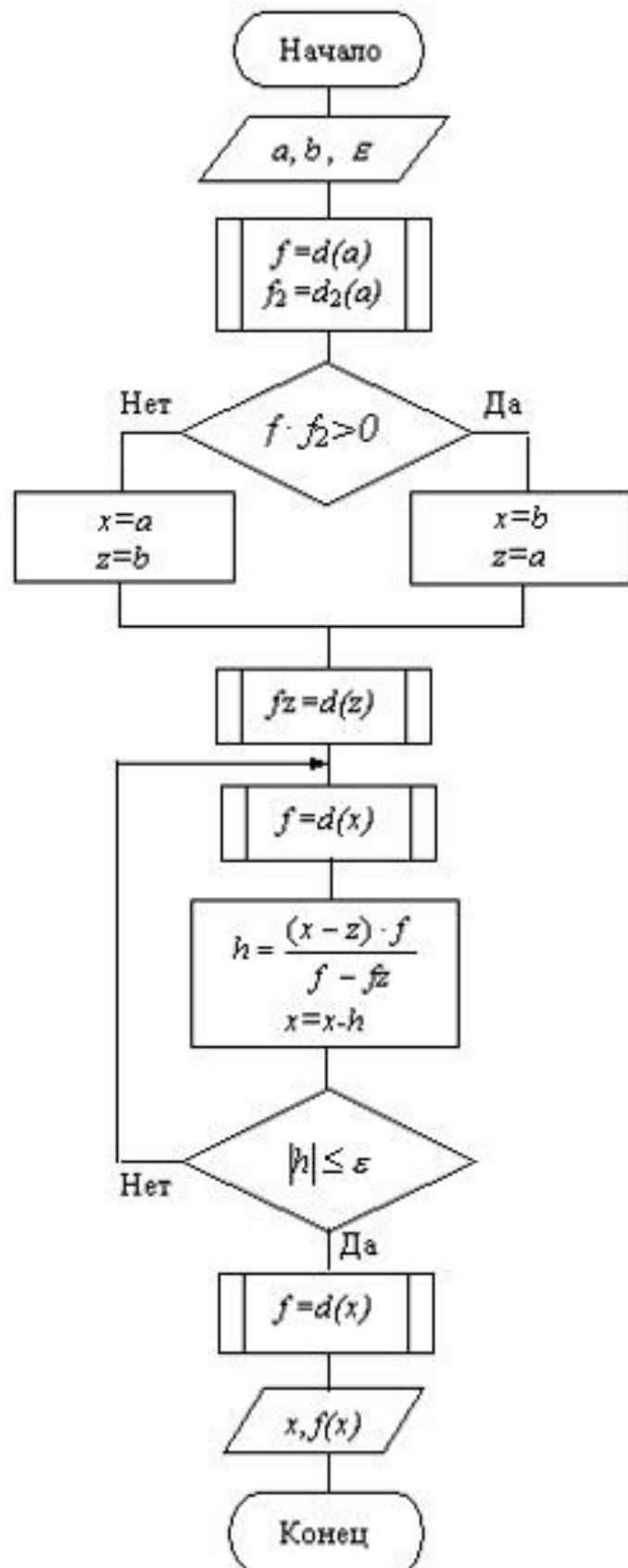


Рис. 4. Схема алгоритма уточнения корня методом хорд

Другим, также распространенным, является метод золотого сечения. Поиск минимума этим способом осуществляется следующим образом. Вычисляется функция на концах отрезка, а также в двух внутренних точках x_1 и x_2 . Сравнивая все четыре значения функции между собой, находят наименьшее. Пусть наименьшее оказалось $f(x_1)$. Очевидно, минимум расположен в одном из прилегающих к нему отрезков. Поэтому отрезок x_2 в отбрасывается, и остается отрезок a x_2 . Первый шаг процесса сделан. На отрезке a x_2 выбирается лишь вторая внутренняя точка, т. к. первая x_1 уже известна из предыдущего шага. Это вчетверо уменьшает объем вычислений на одном шаге.

Как известно, длину шага x_2 x_3 целесообразно выбирать таким образом, чтобы следующий отрезок был поделен подобно предыдущему. Это достигается при длине шага, равном приблизительно 0,38 длины отрезка: $[x_2 x_3] \approx 0,38 [a x_2]$. Таким образом после каждого шага длины отрезка, на котором имеется минимум, сокращаются до 0,62 его длины на предыдущем шаге.

Рассмотренные методы отыскания минимума функций являются достаточно экономичными, они могут быть применены даже к не дифференцируемым функциям и всегда сходятся. Однако сходимость этих методов линейная и относительно медленная.

Если на интервале поиска несколько локальных минимумов, то процесс сойдется к одному из них (но не обязательно к наименьшему).

Рассмотренные алгоритмы после несложных преобразований могут быть использованы для нахождения максимумов функций.

При отыскании экстремумов рассмотренными методами в стохастических задачах необходимо помнить, что из-за ошибок экспериментов можно неправильно определить соотношения между значениями функций в точках, близких к экстремуму. Вблизи экстремума небольшая погрешность функций может привести к появлению довольно большой области неопределенности. Поэтому, если различия функций в выбранных точках стали того же порядка, что и ошибки эксперимента, то итерации надо прекращать.

Основные трудности многомерных задач удобно рассмотреть на примере функции двух переменных $t(x,y)$. Она описывает некоторую поверхность в трехмерном пространстве с координатами x , y , z .

Анализируя формы поверхностей в трехмерном пространстве можно выделить несколько различных типов рельефа (например: котловинный, овражный, неупорядоченный), однако в малой окрестности невыраженного минимума, рельеф таких функций - котловинный (рис. 5).

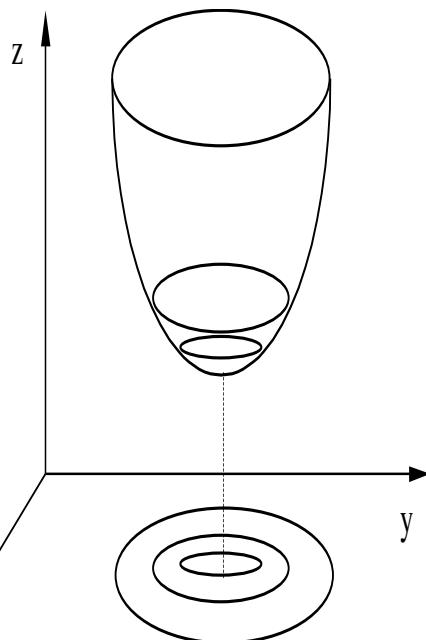


Рис. 5. Котловинный рельеф функции

Для решения заданной задачи отыскания экстремума функции нескольких переменных можно применить рассмотренные выше методы простых итераций, однако в подобных задачах минимизации эти методы обычно сходятся медленно.

Широкое применение для численного поиска экстремума функции нескольких переменных находят методы "спуска", обеспечивающие ускорение итерационного процесса. В данных методах каждое очередное приближение $\bar{X}^{(k+1)}$ (\bar{X} - координатный вектор) получается из предыдущего $\bar{X}^{(k)}$ "шагом" в направлении некоторого вектора $\bar{V}^{(k)}$

$$\bar{X}^{(k+1)} = \bar{X}^{(k)} + \lambda_k \bar{V}^{(k)}$$

Выбор вектора $\bar{V}^{(k)}$ определяет различные варианты метода "спуска". Величина шага управляет выбором λ_k . Точность полученного приближения у минимуму обычно оценивается по величине отклонения двух соседних приближений $X^{(k)}$ и $X^{(k+1)}$, например:

$$\Delta_{k+1} = |X^{(k+1)} - X^{(k)}| = \max |X_i^{(k+1)} - X_i^{(k)}|$$

$$\delta_{k+1} = \max \left| \frac{X_i^{(k+1)} - X_i^{(k)}}{X_i^{(k+1)}} \right|$$

или

где X_i - пространственные координаты $X_1=X, X_2=Y, \dots$

Процесс приближения прекращают, если $\Delta_k < \varepsilon$ (абсолютная оценка приближения) или $\delta_k < \varepsilon$ (относительная оценка приближения), где ε - заданная точность.

В практике наиболее часто применяют "спуск" по координатам, наискорейший спуск и модифицированные методы наискорейшего спуска, метод сопряженных направлений и другие.

В методе спуска по координатам после выбора точки начального приближения (точка A_0 на рис.6) фиксируют вначале одну из координат, например $x=x_0$, используя методы простой итерации, находят минимум функции одной переменной $Z(x_0, y)$ в плоскости X_0YZ (точка A_1). Затем для точки A_1 фиксируют другую координату (y_1), а

минимизацию ведут аналогично в плоскости XY_1Z . Указанную процедуру продолжают до достижения заданной точности.

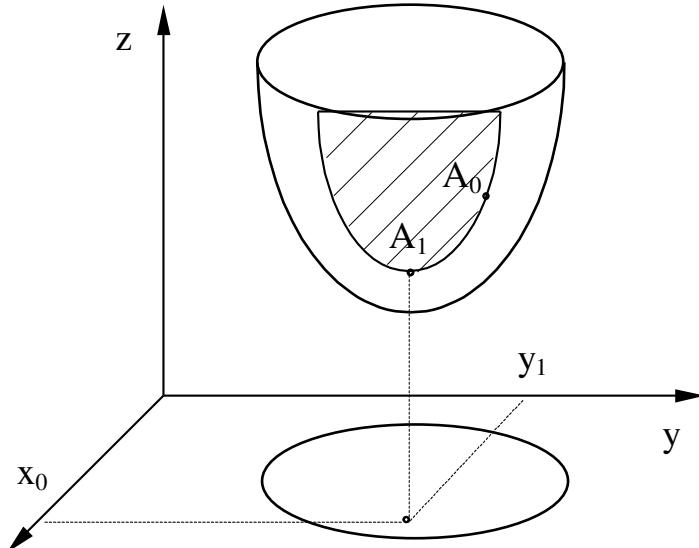


Рис. 6. Спуск по координатам

3. Объект исследования, оборудование

Объектом исследования являются элементы алгоритмического языка C++ и численные методы.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. Варианты заданий

В предлагаемом задании (таблица 1) необходимо численно определить экстремальное значение функции и координаты точки экстремума в заданном интервале поиска одним из рассмотренных методов с абсолютной погрешностью порядка 0.001. На печать вывести вид экстремума (мин., макс.), координаты точки экстремума, соответствующие значения функции, достигнутую погрешность вычислений и число итераций. Необходимо предусмотреть прекращение итерационного цикла, если заданная погрешность не достигается за некоторое максимально допустимое количество повторений. При этом на печать дополнительно должна выводиться надпись “ЗАДАННАЯ ТОЧНОСТЬ НЕ ДОСТИГНУТА”.

Таблица 3.1 Исходные данные

№ вар	Вид экстремума	$x \in [a, b]$		$y \in [c, d]$		Аналитическое решение		
		a	b	c	d	x	y	z
1	MIN	-2	5	0	10	0	3	-9
2	MIN	-4	5	-1	5	0	2	0
3	MAX	0	30	0	40	21	20	282
4	MAX	0,2	0,5	0,3	1	0,25	0,5	1/64
5	MIN	0	1,8	7	0	3,5	-4,8	4,2
6	MIN	4	6	4	8	5	5	-125
7	MAX	-4	4	-6	6	0	0	4
8	MIN	3	5	-4	3	0	0	-3
9	MAX	0	5	-6	0	2,5	-3,1	6,2
10	MIN	-5	5	-8	8	0	0	0
11	MIN	-3	4	-1	5	0	2,1	5,8
12	MAX	-4	4	-5	5	0	0	-2
13	MIN	-6	0	-3	3	-3,7	0	0
14	MIN	1	6	-6	0	3	-3,1	8,2
15	MIN	-3	4	-4	5	0	0	3
16	MAX	1	7	-14	-3	3	9,1	12
17	MIN	1	5	-3	2	3,8	0	0
18	MIN	-3	4	0	5	0	2,1	3,8
19	MAX	0	5	-2	3	3	0	5,1
20	MIN	-3	2	0	5	0	3	1,8
21	MIN	2	3	-3	0	0	-2	0
22	MIN	-2	3	-3	2	0	0	0
23	MIN	-2	3	0	5	0	3,2	5
24	MIN	-3	2	0	5	0	2	0
25	MAX	-3	2	0	5	0	3	9,2

Функции (приложение к таблице).

1) $z = x^2 + xy + y^2 - 3x - 6y$ 14) $z = (x - 3)^2 + (y + 3,1)^2 + 8,2$

2) $z = \sqrt[3]{x^2 + (y - 2)^2} + x^2$ 15) $z = \sqrt{4x^2 + 5y + 9}$

3) $z = \frac{1}{2}xy + (47 - x - y)(\frac{1}{3}x + \frac{1}{4}y)$ 16) $z = 12 - \sqrt[3]{(x - 3)^2 + (y + 9,1)^2}$

4) $z = xy^2(1 - x - y)$ 17) $z = (x - 3,8)^2 e^{-(x^2 - y^2)} + y^2$

5) $z = \sqrt[5]{(x - 3,5)^2 + (y + 4,8)^2} + 4,2$ 18) $z = \sqrt{x^2 + (y - 2,1)^2 + 3,8}$

6) $z = x^3 + y^3 - 15xy$ 19) $z = 5,1 - ((x - 3)^2 + y^2)^{\frac{2}{5}}$

7) $z = 4 - (x^2 + y^2)^{\frac{2}{3}}$ 20) $z = x^2 + (y - 3)^4 + 1,8$

8) $z = \sqrt[3]{(x^2 + y^2)} - 3$ 21) $z = x^4 + (y + 2)^2 + 3,2x^2$

9) $z = 6,2 - \sqrt[3]{(x - 2,5)^2 + (y + 3,1)^2}$ 22) $z = x^2(y + 2)^2 + x^2 + (y + 2)^4$

$$10) z = (x^2 + y^2)e^{-(x^2+y^2)} + (x^2 + y^2)$$

$$23) z = \sqrt[5]{x^2 + (3,2 - y)^2 + 5}$$

$$11) z = 0,5x^2 + 3(y - 2,1)^2 + 5,8$$

$$24) z = y^2 + \sqrt[3]{(x - 2)^2 + y^2}$$

$$12) z = -\sqrt{3,8x^2 + 4,2y^2 + 4}$$

$$25) z = 9,2 - (x^2 + (y - 3)^2)^{\frac{2}{3}}$$

$$13) z = \sqrt[5]{(x + 3,7)^2 + y^2} + y^2$$

5. Порядок выполнения работы

1. Выбрать метод численного поиска экстремума и разработать алгоритм решения задачи.
2. Составить и отладить программу вычисления точки экстремума функций по разработанному алгоритму.
3. Выполнить необходимые расчеты по разработанной программе.
4. Проанализировать результаты расчеты, сравнить с аналитическим решением.
5. Ответить на контрольные вопросы.

6. Указания по оформлению отчета.

Каждый студент оформляет отчет, который должен содержать:

- постановку задачи;
- разработанный алгоритм решения;
- программу расчета;
- тексты распечаток, полученных при отладке программы;
- распечатку результатов расчетов;
- перечень и характеристику ошибок, допущенных в процессе прохождения задания;
- анализ результатов;
- ответ на контрольные вопросы.

Для составленной программы необходимо дать таблицу использованных имен переменных.

7. Контрольные вопросы.

1. Что такое рельеф функции? И как он влияет на сходимость численного поиска экстремума функции?
2. Что будет получено в результате выполнения разработанной вами программы, если в области поиска нет локального экстремума?

ЛАБОРАТОРНАЯ РАБОТА № 6
РЕШЕНИЕ
ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

1. ЦЕЛЬ РАБОТЫ.

Целью данной работы является:

- практика в использовании численных методов решения алгебраических и трансцендентных уравнений;
- практика в разработке алгоритмов и программ, содержащих итерационные циклы;
- получение начальных навыков использования библиотек стандартных программ и личных библиотек;
- углубление навыков отладки программ и практической работы на ЭВМ.

2. ОСНОВЫ ТЕОРИИ.

Аналитическое решение для многих нелинейных алгебраических и трансцендентных уравнений получить не удается. Общая форма таких уравнений $f(x) = 0$.

Алгебраические уравнения $a_{m+1}x^m + a_m x^{m-1} + a_{m-1} x^{m-2} + \dots + a_1 = 0$ имеют m корней. Трансцендентные уравнения, включающие степенные, тригонометрические и экспоненциальные функции от некоторого аргумента X , например $\arctg(x) - x = 0$, бесчисленное множество корней. Для решения таких уравнений используют приближенные итерационные методы. Решение уравнения обычно складывается из двух этапов: отыскание начального приближения корня, т.е. определение интервалов, в которых имеется корень уравнениями последующего уточнения начального приближения корня до достижения заданной точности.

Процесс определения интервала, содержащего только один из корней уравнения, называется отделением корня этого уравнения. Обычно процесс отделения корней проводят исходя из физического смысла задачи, графически или с помощью таблиц значений функций $f(x)$. Известно, что если непрерывная функция $f(x)$ на концах отрезка $[a, b]$ принимает значения разных знаков, то внутри этого отрезка есть хотя бы одна точка $x=u$, в которой функция принимает нулевое значение $f(u)=0$. Если при этом знак первой производной $f'(x)$ на этом отрезке не изменяется, то корень $x=u$ является единственным на данном отрезке.

Наиболее распространенным численными методами уточнения корней являются методы последовательного приближения, половинного деления, Ньютона, итерации.

Метод половинного деления

Уточнение корня производится в следующей последовательности:

- вычисляется координата x_1 середины отрезка поиска $[a, b]$;
- определяются знаки функции $f(x)$ в точках x, a, b ;
- определяется новый уменьшенный интервал поиска по результатам сравнения знаков функции $f(x)$ в указанных точках (отбрасывается та половина предыдущего интервала, которая содержит на границе значения функции $f(x)$ того же знака, что и в середине интервала);
- указанная последовательность действий повторяется до достижения требуемой точности $|x_j - x_{j-1}| < \varepsilon$, где ε - допустимая погрешность решения.

Применение метода половинного деления проиллюстрировано на рис.1.

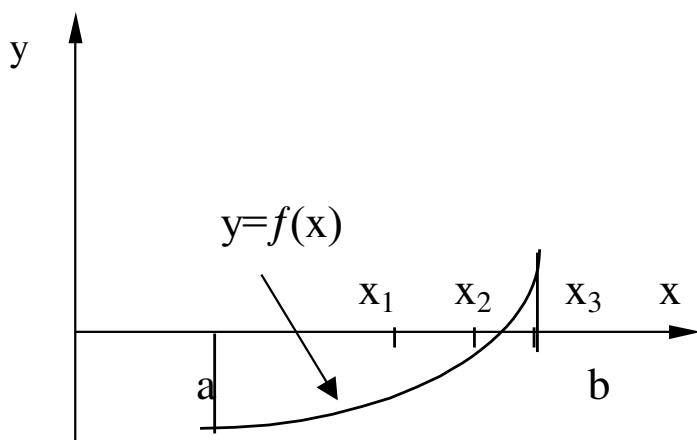
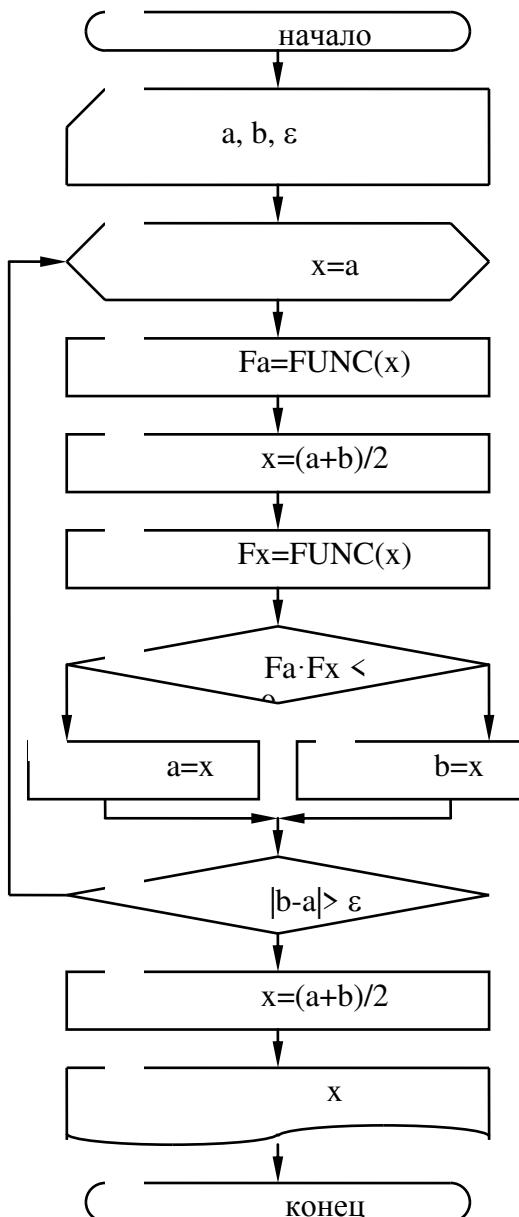


Рис.1. Уточнение корня методом половинного деления

Алгоритм рассмотренного вычислительного процесса имеет вид:



Метод Ньютона

Уточнение корня может быть произведено также по методу Ньютона. Сущность метода Ньютона заключается в том, что в интервале поиска выбирается начальное приближение корня x_0 (рис.2) и в этой точке проводится касательная к функции, и точка x_1 пересечения касательной с осью абсцисс принимается за уточненное значение корня.

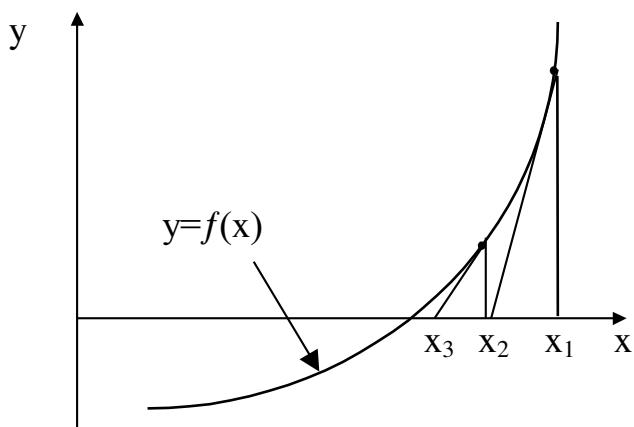


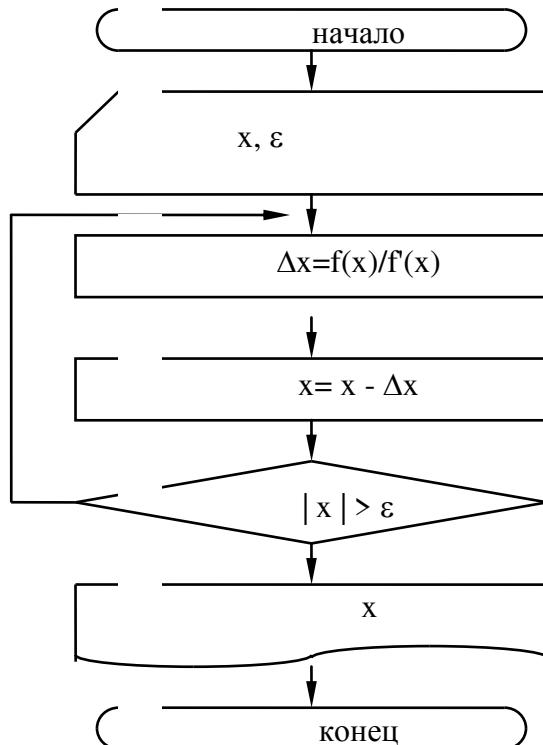
Рис. 2. Графическая интерпретация метода Ньютона

Повторяя построение касательных в точках $x_1, x_2, x_3 \dots x_{n-1}, x_n$, получают последовательно уточнение корня. Аналитическая зависимость, описывающая такой процесс, имеет вид:

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

Метод Ньютона (касательных) в отличие от метода половинного деления использует информацию о форме функции, что ускоряет процесс уточнения корня. Однако данный метод ограничен в применении, поскольку для функций с изменением кривизны и пологими участками в интервале поиска пересечение касательной с осью абсцисс может выйти за пределы интервала, и тогда уточнения корня не получится.

Алгоритм уточнения корня по методу Ньютона имеет вид:

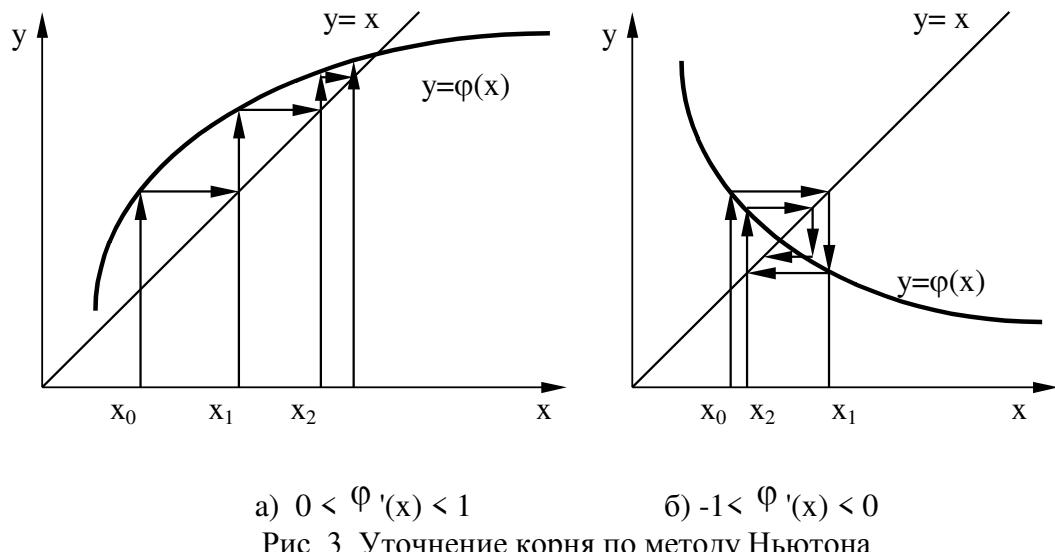


Метод простой итерации

Метод простой итерации (последовательных приближений) заключается в том, что исходное уравнение имеет вид $\Phi(x) = x$. Если в интервале поиска выполняется условие $|\Phi'(x)| < 1$, то метод дает возможность вычислить значение корня с заданной точностью. Если это условие не выполняется, то можно перейти к обратной функции. Приближение к корню осуществляется по формуле $x_{j+1} = \Phi(x_j)$. Итерационный процесс прекращается, если выполняется условие

$$|x_{i+1} - x_i| \leq \varepsilon$$

где ε - допустимая погрешность решения. Сходимость будет тем более быстрой, чем меньше величина $|\Phi'(x)|$.



a) $0 < \Phi'(x) < 1$

б) $-1 < \Phi'(x) < 0$

Рис. 3. Уточнение корня по методу Ньютона

Следует отметить, что для всякого уравнения $f(x) = 0$ можно найти большое количество соответствующих ему уравнений $x = \varphi(x)$, но нужно с большой осторожностью подходить к их конкретному выбору, т.к. от него зависит сходимость и скорость сходимости метода итераций.

Приближенное решение систем нелинейных алгебраических и трансцендентных уравнений

Приближенное решение систем нелинейных алгебраических и трансцендентных уравнений

$$f_1(x_1, x_2, \dots, x_n) = 0;$$

$$f_2(x_1, x_2, \dots, x_n) = 0;$$

.....

$$f_n(x_1, x_2, \dots, x_n) = 0;$$

также осуществляется в два этапа: отделение корней и уточнение корней с помощью метода последовательных приближений (методом Ньютона или методом итераций). Однако при уточнении корней, систем уравнений в форме $x_j = \Phi(x_1, x_2, \dots, x_n)$ представление их и анализ сходимости процесса итераций более трудоемки и сложны. Изменение формы исходного уравнения при этом неоднозначно поэтому необходимо

тщательно проанализировать различные варианты преобразованных уравнений с целью получения пригодной для итерации формы.

В заключении необходимо отметить, что допустимую погрешность ε определения корня уравнения в итерационном процессе нельзя задавать слишком малой, т.к. ошибки округления в ЭВМ не позволяют получить более точного приближения.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы численных методов и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

Для функции, заданной в каждом варианте, необходимо найти двумя различными приближенными методами наименьший по модулю отличный от нуля корень уравнения с относительной погрешностью не более $\varepsilon=0,001$. Три шага приближения по каждому из методов выполнить вручную с помощью микрокалькулятора и изобразить графически.

- 1) $1/(1+x^2) - 1,5x = 0$
- 2) $0,1x^2 - x \ln(x) = 0$
- 3) $x^3 - 1,473x^2 - 5,738x + 6,763 = 0$
- 4) $\operatorname{tg}^2 x - 1,5x = 0$
- 5) $e^{-x} - 1,5x = 0$
- 6) $1/(1+x^4) - 1,5x^2 = 0$
- 7) $\ln(2+x) - 5,5x^3 = 0$
- 8) $x^3 - 10 - 1,5\sqrt{x-2} = 0$
- 9) $\sqrt{1-x^2} - 2,5x^5 = 0$
- 10) $1-x^2 - 0,4e^x = 0$
- 11) $\sin 2x - 2x^2 = 0$
- 12) $2x - e^{-x/10} = 0$
- 13) $e^{-0,3x} = 0,7x$
- 14) $x^3 - 3x - 1 = 0$
- 15) $\sin x - x \cos x = 0$
- 16) $x^3 + 2x^2 - 10,2x = 0$
- 17) $x = \operatorname{tg} x$
- 18) $x^4 - 2,5x^2 + x = 0$
- 19) $1/(1+x^2) - 2,5x^2 = 0$
- 20) $x^3 + 3x + 1 = 0$
- 21) $1,5\cos x = 2x^2$
- 22) $4x^3 - 12,3x^2 - x + 16,2 = 0$

$$23) \ln(1,5x + 3,2) = 4,3x$$

$$24) 2,5x^3 + 1,2x^2 = 3,2$$

$$25) 1,2e^{-x} = \cos x$$

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

- Исследовать заданную функцию, найти интервал, в котором находится требуемый корень уравнения, проверить применимость различных численных методов и выбрать метод решения.
- Разработать алгоритм решения задачи двумя выбранными методами, представив его структуру в виде блок - схемы и дав его неформальное описание.
- Составить и отладить программу решения задачи.
- Вычислить три шага приближения вручную и построить график приближения.
- Составить программу решения задачи с помощью одной из стандартных подпрограмм.
- Решить задачу на ЭВМ по разработанным программам.
- Проанализировать результаты расчетов.
- Ответить на контрольные вопросы.

При отладке программы следует прежде всего отладить используемую подпрограмму вычисления функции. Для этого нужно подготовить и решить соответствующий набор тестов.

Отладку численного решения уравнения целесообразно сначала провести на уравнении с заранее известным решением.

6. УКАЗАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Каждый студент оформляет отчет, который должен содержать:

- постановку задачи;
- алгоритмы решения задачи;
- программы расчета;
- тексты распечаток отладок программ;
- распечатки результатов;
- перечень и характеристика ошибок, допущенных в процессе прохождения задания;
- ответы на контрольные вопросы.

К составленным программам следует дать таблицу использованных имен переменных.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем определяется существование предела достижимой точности приближенного вычисления корней, одинаков ли этот предел для различных методов?

- 2.Каким образом можно предусмотреть выход из итерационного процесса, если заданная точность не достигается?
- 3.Какое влияние на конечный результат вычисления корня уравнения в итерационном процессе оказывает ошибка, допущенная на промежуточном шаге данного процесса?

ЛАБОРАТОРНАЯ РАБОТА № 7

ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

1 ЦЕЛЬ РАБОТЫ

Целью данной работы является:

- практика в использовании численных методов интегрирования функций на ЭВМ;
- углубление навыков разработки алгоритмов и программ, имеющих модульную структуру;
- практика в использовании библиотеки стандартных подпрограмм и личных библиотек.

2. ОСНОВЫ ТЕОРИИ

Задачи, в которых требуется вычисление определенных интегралов, возникают почти во всех областях прикладной математики. Иногда можно вывести аналитическую формулу и представить интеграл в виде комбинации алгебраических и трансцендентных функций с соответствующими пределами. Во многих случаях однако, не удается найти никакой аналитической зависимости или же она получается настолько сложной, что вычислить с ее помощью интеграл труднее, чем другими способами. В таких случаях приходится применять различные методы численного интегрирования, которые основаны на том что интеграл представляется в виде конечной суммы простых слагаемых. В геометрической интерпретации при численном интегрировании площадь под кривой интегрирования приближенно заменяется суммой площадей элементарных фигур (прямоугольников, трапеций и др.), которые могут быть найдены по простым аналитическим зависимостям.

Наиболее распространенными методами численного интегрирования функций на ЭВМ является метод прямоугольников, частным случаем которого является метод средних, метод трапеций и метод Симпсона (метод парабол). Указанные методы различаются способом аппроксимации интегрируемой функции на каждом шаге интегрирования. В методе прямоугольников применяется ступенчатая аппроксимация, в методе трапеций – линейная аппроксимация, в методе Симпсона – аппроксимация параболой второй степени.

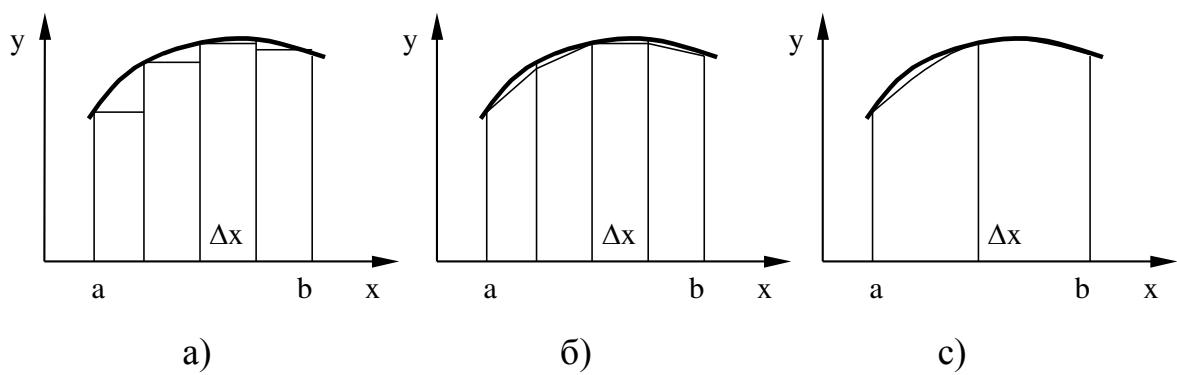


Рис. 1. Геометрическое представление численных методов интегрирования:

- a) - метод прямоугольников;
- б) - метод трапеций;
- в) - метод Симпсона.

Указанные численные методы могут применяться не только к функциям, заданным аналитически, но и к табличным функциям, широко распространенной в инженерной практике (это результаты экспериментов, справочные таблицы свойств материалов и т.д.)

Квадратурные формулы указанных методов для постоянного шага интегрирования Δx представлены ниже.

1. Метод прямоугольников

$$S = \int_a^b f(x) dx \approx \Delta x (f(x_1) + f(x_2) + \dots + f(x_n)),$$

$\Delta x = (b-a)/n$, n — число шагов интегрирования

Остаточные члены для различных вариантов метода прямоугольников вычисляются формулами:

$$x_i = a + \Delta x * (i-1) : R = -(\Delta x)^2 / 2 \cdot f'(x_i) = 0 [(\Delta x)^2];$$

$$x_i = a + \Delta x * i : R = (\Delta x)^2 / 2 \cdot f'(x_i) = 0 [(\Delta x)^2];$$

$$x_i = a + \Delta x * (i-1/2) : R = (\Delta x)^3 / 2a \cdot f''(x_i) = 0 [(\Delta x)^3];$$

2. Метод трапеций.

$$S = \int_a^b f(x) dx \approx \Delta x \cdot (1/2 f(x_1) + f(x_2) + f(x_3) + \dots + f(x_n) + 1/2 f(x_{n+1}))$$

где $x_i = a + \Delta x(i-1)$, $\Delta x = (b-a)/n$

Остаточный член: $R = -(\Delta x)^3 / 12 f'''(x_i) = 0 [(\Delta x)^3]$

3. Формула Симпсона.

$$S = \int_a^b f(x) dx \approx \Delta x / 3 * (f(x_1) + 4f(x_2) + 2f(x_3) + \dots + 4f(x_n) + f(x_{n+1})),$$

где $x_i = a + \Delta x * (i-1)$, $\Delta x = (b-a)/n$.

Остаточный член: $R = (\Delta x)^4 / 180 f''''(x_i) = 0 [(\Delta x)^4]$

Для обеспечения требуемой точности при приближенном вычислении значения интеграла по квадратурным формулам на практике часто используется метод последовательного удвоения числа шагов, который заключается в следующем.

Интеграл S вычисляется по квадратурной формуле дважды: сначала при числе шагов, равном n , а затем при числе шагов равно $2n$.

Погрешность приближенного значения интеграла S_n , вычисленного по квадратурной формуле при числе шагов, равном $2n$, определяется приближенно по правилу Рунге:

$$\varepsilon_{2n} \approx \theta \cdot |S_n - S_{2n}|,$$

где для формулы средних и трапеций $\theta=1/2$, для формулы Симпсона $\theta=1/16$.

Таким образом, S_n вычисляется для последовательных значений n , $2n$, $4n$ и т. д. Процесс вычисления заканчивается, когда для очередного числа шагов интегрирования не будет получена допустимая погрешность. Начальное число шагов n рекомендуется выбирать от 10 до 100.

Для экономии следует учесть, что при удвоении числа шагов нет необходимости заново вычислять значения подынтегральной функции во всех узлах, так как часть узлов сетки с шагом $2n$ являются узлами и ранее, и в них уже вычислялись значения функции.

Следует так же учитывать, что знаки погрешностей у формулы средних и формулы трапеций разные. Поэтому, если есть расчеты по обеим формулам, то можно утверждать, что точное значение интеграла, как правило, лежит между этими результатами.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы численных методов и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

С использованием ЭВМ вычислить с относительной погрешностью не

$$\int_a^b f(x)dx$$

более 10^{-4} значение для исходных данных, приведенных в таблице 1. Вычисление произвести по разработанной самостоятельно программе и с использованием одной из стандартных библиотечных подпрограмм, исследовать влияние числа шагов интегрирования на точность решения задачи. Разработанную программу представить в виде подпрограммы и записать ее в личную библиотеку исходных модулей, обеспечив возможность интегрирования функции, задаваемой

пользователем во внешней подпрограмме – функции предусмотрите возможность вывода вместе с величиной интеграла – числа шагов интегрирования (величину шагов интегрирования), при котором достигнута заданная точность.

Таблица 1. Исходные данные

№ п/п	f(x)	a	b	Квадратурная формула
1	$\sqrt{e^x - 1}$	0,1	2	прямоугольников
2	$e^x \cdot \sin(x)$	0	π	трапеций
3	$(x^2 - 1) \cdot 10 - 2x$	0	1	средних
4	$x \cdot \sqrt[3]{1 + x}$	1	9	Симпсона
5	$1/[3+2\cos(x)]$	0	π	прямоугольников
6	$1/(\ln 2x)$	2	3	трапеций
7	$\arcsin(\sqrt{x} / \sqrt{x(1-x)})$	0	0,3	средних
8	$x^3 e^2 x$	0	1	Симпсона
9	$\operatorname{tg}[3(x/2+\pi/4)]$	0	$\pi/4$	прямоугольников
10	$\operatorname{arctg}(x)$	0	$\sqrt{3}$	трапеций
11	$1/(1 + \sqrt{x})$	0	4	средних
12	$1/[5-3\cos(x)]$	0	2π	Симпсона
13	$2x/(1-4x)$	-2	-1	прямоугольников
14	$1/[(x+1)\sqrt{x^2 + 1}]$	0	$3/4$	трапеций
15	$\cos(\pi x^2) / \sqrt{1,2 + x^3}$	0,6	0,7	средних
16	$(0,3+x^2)\cos \sqrt{e-x}$	0,2	0,6	Симпсона
17	$\sin(ex^2) / \sqrt{0,8 + 3x}$	0,5	0,6	прямоугольников
18	$e^{-x}/4/(\pi+x^2)$	0	1	трапеций
19	$\sqrt{1,5 - 0,4x} \operatorname{tg}(x/2/5\pi)$	1,5	2	средних
20	$e^{-0,3x} / \sqrt{2\pi + x}$	0,5	1,5	Симпсона

21	$(\pi - x^2) \sin \sqrt{2,1+x}$	5	25	прямоугольников
22	$\sqrt{0,7 - 0,1x} \ln[0.3(\pi + x)]$	-1	4	трапеций
23	$\sqrt[3]{0,1 + 0,2x} \ln(\pi/3 - x/5)$	-3	2	средних
24	$(2,5-x)\operatorname{ctg}(\pi/x)$	1,8	2,3	Симпсона
25	$1,5/\sqrt[4]{x^2 + 2}$	0	2,1	прямоугольников

5 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

1. Разработать алгоритм решения задачи, представив его структуру в виде блок-схемы и дав его неформальное описание.
2. Составить и отладить программу решения задачи по разработанному алгоритму.
3. Записать разработанную программу в личную библиотеку.
4. Составить программу решения задачи с помощью одной из стандартных программ.
5. Решить задачу на ЭВМ по разработанным программам.
6. Проанализировать результаты расчетов, построить график зависимости точности решения от числа шагов интегрирования.
7. Ответить на контрольные вопросы.

Отладку разработанной программы провести на несложном уравнении с заранее известным решением.

6. УКАЗАНИЕ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Каждый студент оформляет отчет, который должен содержать :

- постановку задачи;
- алгоритм решения задачи;
- программы расчета;
- тексты распечаток отладки программы;
- распечатки результатов расчетов;
- перечень и характеристика ошибок, допущенных в результате прохождения заданий;
- график зависимости точности решения от числа шагов интегрирования;
- ответы на контрольные вопросы.

К составленным программам следует дать таблицу использованных имен переменных.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

Существует ли предел достижимой точности вычисления интеграла рассмотренными методами?

Каковы преимущества формулы Симпсона по сравнению с формулой трапеций и следствием чего являются эти преимущества?

Почему остаточные члены формулы средних и формулы трапеций имеют разные знаки? Дайте геометрическую интерпретацию ответа и поясните, как можно использовать указанное обстоятельства при расчетах. Какие другие формулы численного интегрирования функций с разными знаками остаточных членов вы знаете?

ЛАБОРАТОРНАЯ РАБОТА № 8

РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ МЕТОДОМ ЭЙЛЕРА

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является:

- овладение практическими навыками использования ЭВМ для численного решения обыкновенных дифференциальных уравнений первого порядка (задача Коши);
- проведение вычислительных экспериментов по применению метода Эйлера;
- изучение влияния шага интегрирования на точность решения конкретного уравнения ;
- накопление опыта по использования алгоритмических языков и отладке программ;

2. ОСНОВЫ ТЕОРИИ

Одним из наиболее распространенных методов решения обыкновенных дифференциальных уравнений является метод конечных разностей (МКР). Рассмотрим применение МКР для численного решения на ЭВМ простейшего дифференциального уравнения первого порядка:

$$\frac{dY}{dX} = f(X, Y)$$

с начальными условиями X_0 , $Y(X_0) = Y_0$.

Решение будем искать в интервале $[X_0, b]$ и будем полагать, что функция на данном интервале удовлетворяет условиям гладкости.

Разобьем область аргумента X на множество отрезков длиной ΔX и разложим функцию Y в ряд Тейлора в окрестности произвольной точки X_i из области существования функции:

$$Y_{i+1} = Y(X_i + \Delta X) = Y(X_i) + \frac{\Delta X}{1!} Y'(X_i) + \frac{\Delta X^2}{2!} Y''(X_i) + \dots$$

Отбрасывая члены ряда, содержащие производные второго и высшего порядков, получаем конечно-разностное выражение первой производной

$$Y'(X_i) = \frac{Y_{i+1} - Y_i}{\Delta X}.$$

Отсюда $Y_{i+1} = Y_i + \Delta X * f(X_i, Y_i)$.

Вычисляя последовательно от начального значения Y_0 значения Y_1 , Y_2 , Y_3 , ..., Y_{i+1} по данной формуле, находим искомое решение.

На рис.2 показана форма численного решения, получаемая с помощью таких вычислений.

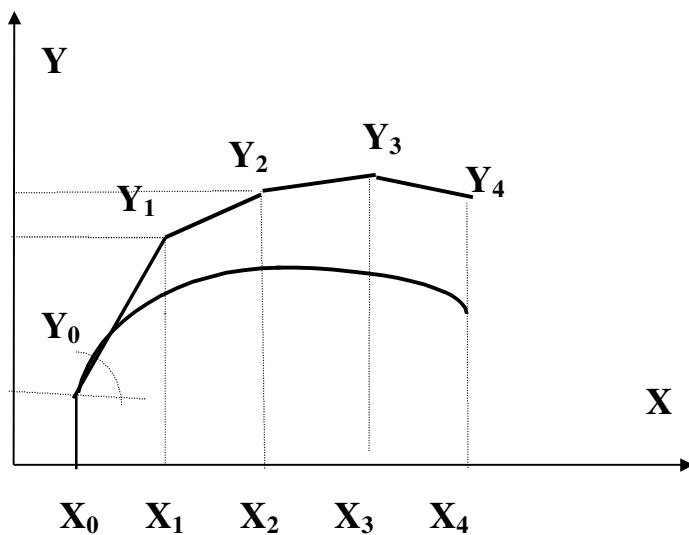
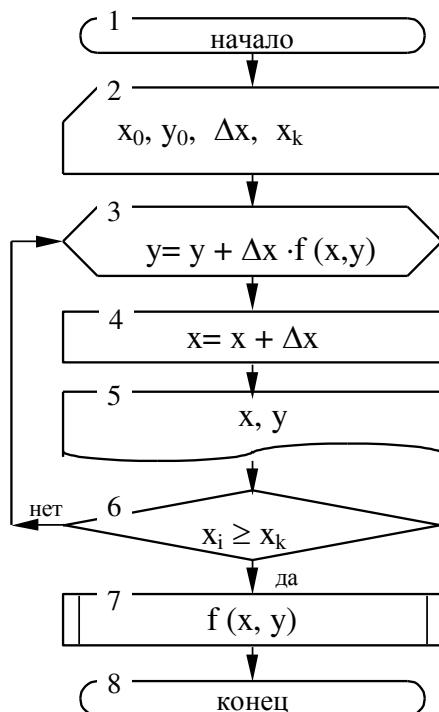


Рис.2. Схема приближенного решения методом Эйлера

Данный метод решения обыкновенного дифференциального уравнения носит название метода Эйлера. При достаточно малых величинах шага ΔX метод Эйлера дает решение с большой точностью, так как погрешность близка к 0 (ΔX^2) на каждом шаге процесса по методу Эйлера.



В данной блок-схеме x_k - конечное значение координаты X

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы численных методов и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

Дано дифференциальное уравнение первого порядка

$$Y'(X) = f(X, Y),$$

где $f(X, Y)$ - заданная функция.

Требуется найти численное решение задачи Коши на заданном отрезке $[X_0, b]$ ($b > X_0$) при начальном условии $Y(X_0) = Y_0$ методом Эйлера с различной величиной шага интегрирования ΔX и исследовать влияние величины шага интегрирования на точность решения (сравнение осуществлять с аналитическим решением задачи).

Ниже приведены варианты функций $f(X, Y)$.

Функция может быть задана произвольным набором двух функций Φ , Ψ (X, Y), Ψ (X, Y):

$$f(X, Y) = \Phi(X, Y) + \Psi(X, Y).$$

Функции Φ (X, Y):

- | | | |
|----------------|----------------|------------------|
| 1) Y / X ; | 2) $X^2 + Y$; | 3) $X^3 + Y$; |
| 4) $X + Y^2$; | 5) $X + Y^3$; | 6) $X^2 + Y^2$; |
| 7) $e^x Y$; | 8) e^{x+1} ; | 9) $e^x + Y^2$. |

Функции Ψ (X, Y):

- | | | |
|--------------------|------------------|---------------------|
| 1) $e^{-x^2} XY$; | 2) e^{x^2+1} ; | 3) XY ; |
| 4) 3.2; | 5) $4.8X^2Y$; | 6) $e^x + 3Y$; |
| 7) $e^{3x} Y$; | 8) 8.7; | 9) $\sqrt{X} + Y$. |

Решение задачи осуществить в интервале $[2, 3]$ при начальном условии, заданном преподавателем.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

1. Изучение метода Эйлера.
2. Составление блок-схемы алгоритма.
3. Составление программы расчета (в программе предусмотреть возможность ее использования для различных шагов интегрирования ΔX).
4. Ввод программы в ЭВМ и отладка программы.
5. Проведение вычислительных экспериментов.

6.Анализ полученных результатов.

6.УКАЗАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

- постановку задачи;
- блок - схему алгоритма;
- распечатки отладок программы;
- распечатки результатов решения
- анализ результатов;
- перечень ошибок , обнаруженных в процессе отладки;
- ответы на конкретные вопросы.

В отчете следует дать необходимые пояснения использованных в программе имен переменных.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем ограничена минимальная величина шага интегрирования ΔX ?
2. Каким образом можно уточнить решение дифференциального уравнения методом Эйлера, не изменяя величину шага интегрирования ΔX ?
3. Какое изменение необходимо ввести в ваш алгоритм , чтобы получить решение с заданной точностью?

ЛАБОРАТОРНАЯ РАБОТА № 9

РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ МЕТОДОМ РУНГЕ-КУТТА

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является:

- овладение практическими навыками численного решения задач Коши для обыкновенных дифференциальных уравнений;
- проведение вычислительных экспериментов по применению метода Рунге-Кутта;
- сравнение методов Эйлера и Рунге-Кутта;
- практика и использование стандартных подпрограмм для решения дифференциальных уравнений;
- закрепление навыков программирования на алгоритмическом языке и отладки программ.

2. ОСНОВЫ ТЕОРИИ

Увеличение точности решения при укрупненных шагах интегрирования обеспечивают методы Рунге-Кутта. Уточнение достигается за счет специального подбора координат промежуточных на шаге интегрирования точек, в которых вычисляется первая производная. Вместо значения первой производной в начале шага интегрирования, используемой в методе Эйлера, вычисляется усредненная на шаге интегрирования первая производная. Формула численного интегрирования приобретает вид:

$$Y_{i+1} = Y_i + \Delta X * \tilde{f},$$

где \tilde{f} - усредненная на шаге интегрирования первая производная .

Ниже приведены формулы метода Рунге-Кутта различного порядка:

1. Метод 2-го порядка, $\epsilon = 10^{-2}$:

- первый вариант $Y_{i+1} = Y_i + K_2$,

где $K_2 = \Delta X * f(X_i + \Delta X / 2, Y_i + K_1 / 2)$,

$K_1 = \Delta X * f(X_i, Y_i)$;

- второй вариант $Y_{i+1} = Y_i + 1/2 (K_1 + K_2)$,

где $K_1 = \Delta X * f(X_i, Y_i)$,

$K_2 = \Delta X * f(X_i + \Delta X, Y_i + K_1)$.

2. Метод 3-го порядка , $\epsilon = 10^{-3}$:

- первый вариант $Y_{i+1} = Y_i + 1/4K_1 + 3/4K_3$,

где $K_1 = \Delta X * f(X_i, Y_i)$,

$K_2 = \Delta X * f(X_i + 1/3 * \Delta X, Y_i + 1/3K_1)$,

$K_3 = \Delta X * f(X_i + 2/3 * \Delta X, Y_i + 2/3K_2)$;

- второй вариант $Y_{i+1} = Y_i + 1/6 (K_1 + 4K_2 + K_3)$,
где $K_1 = \Delta X * f(X_i, Y_i)$,
 $K_2 = \Delta X * f(X_i + 1/2 * \Delta X, Y_i + 1/2 K_1)$,
 $K_3 = \Delta X * f(X_i + \Delta X, Y_i - K_1 + 2K_2)$;

3. Метод 4-го порядка, $\varepsilon = 10^{-4}$:

- первый вариант $Y_{i+1} = Y_i + 1/6 (K_1 + 4K_3 + K_4)$,
где $K_1 = \Delta X * f(X_i, Y_i)$,
 $K_2 = \Delta X * f(X_i + 1/4 * \Delta X, Y_i + 1/4 K_1)$,
 $K_3 = \Delta X * f(X_i + 1/2 * \Delta X, Y_i + 1/2 K_2)$,
 $K_4 = \Delta X * f(X_i + \Delta X, Y_i + K_1 - 2K_2 + 2K_3)$;
- второй вариант $Y_{i+1} = Y_i + 1/6 (K_1 + 2K_2 + 2K_3 + K_4)$,
где $K_1 = \Delta X * f(X_i, Y_i)$,
 $K_2 = \Delta X * f(X_i + 1/2 * \Delta X, Y_i + 1/2 K_1)$,
 $K_3 = \Delta X * f(X_i + 1/2 * \Delta X, Y_i + 1/2 K_2)$,
 $K_4 = \Delta X * f(X_i + \Delta X, Y_i + K_3)$.

В практических расчётах интегрирование дифференциальных уравнений осуществляется с автоматическим выбором шага, обеспечивающее получение результата с заданной погрешностью решения.

При интегрировании с автоматическим выбором шага рекомендуется использовать следующие правила выбора шага .

В узле X_0 взять $\Delta X = \Delta X_0$, ΔX_0 – заданный начальный шаг, найти приближённые значения решения \tilde{Y} и $\tilde{\tilde{Y}}$ с шагами ΔX и $\Delta X/2$ соответственно. За абсолютную погрешность приближённого решения (в качестве которого естественно взять $\tilde{\tilde{Y}}$ как более точное), вычисленного по методу Рунге-Кутта n -го порядка (метод Эйлера является методом Рунге-Кутта первого порядка), принимается

$$\delta = \left| \frac{\tilde{Y} - \tilde{\tilde{Y}}}{2^n - 1} \right|,$$

Если $\delta \geq \varepsilon$, то шаг ΔX уменьшается в два раза и вычисления повторяются, исходя из узла X_0 . Как только на очередном приближении будет получено $\delta < \varepsilon$, считается, что $\tilde{\tilde{Y}}$ и является решением в узле $X_1 = X_0 + \Delta X$, полученным с заданной точностью на этом шаге .

Решение в следующем узле X_2 , исходя из узла X_1 , получается аналогичным образом. При этом начальный шаг выбирается по шагу ΔX , с которым было получено решение в узле X_1 , в зависимости от погрешности δ : если $\delta < \varepsilon / 2^{n+1}$, то предыдущий шаг удваивается; в противном случае шаг не изменяется. Аналогично находится решение и в последующих точках .

При выходе к точке $X = b$ следует проявлять осторожность, т.к. при значениях $X > b$ правая часть $f(X, Y)$ дифференциального уравнения может быть не определена и при решении задачи на ЭВМ может произойти прерывание, в результате чего задача будет снята со счёта.

Во избежание такой ситуации необходимо на каждом шаге интегрирования проверять условие выхода за пределы интервала интегрирования. На шаге выхода за точку $X = b$ принять последнее значение ΔX таким, чтобы точно выйти на точку b .

Для отладки программы можно взять простейшее уравнение и небольшой отрезок интегрирования.

В библиотеке стандартных программ имеются программы, для интегрирования дифференциальных уравнений методом Рунге-Кутта.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы численных методов и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

Дано дифференциальное уравнение первого порядка вида

$$Y'(X) = f(X, Y) = F(X) - g(X)*Y(X),$$

где $F(X)$, $g(X)$ – заданные функции.

Требуется:

1. Найти численное решение задачи Коши на заданном отрезке $[X_0, b]$ при начальном условии $Y(X_0) = Y_0$ одним из методов Рунге-Кутта:

а) с постоянным шагом $\Delta X = (b - X_0) / N$, где N – число шагов;

б) с автоматическим выбором шага, выйдя на точку $X = b$ с точностью $\epsilon = 10^{-4}$.

2. Найти решение поставленной задачи с использованием стандартной подпрограммы.

3. Сравнить численные решения, полученные методами Рунге-Кутта с постоянным шагом, с автоматическим выбором и с помощью стандартной программы.

Ниже приведены варианты функций $g(X)$ и $F(X) = \Phi(X)*\Psi(X)$. Функция $F(X)$ может быть задана произвольным набором функций $\Phi(X)$ и $\Psi(X)$ в заданном интервале.

3.1. $\Delta X_0 = 0.5$, $X_0 = 1$, $b = 6$, $Y_0 = 10$, $g(X) = 2(X - 2)$.

Функции $\Phi(X)$:

$$1) e^{-(x+4)x} \quad 2) e^{-x^2}; \quad 3) e^{-x^2+2x};$$

- 4) $x e^{-x^2}$; 5) $(x+2)e^{-x^2}$; 6) $e^{-x^2} \sin x$;
 7) $e^{-x^2} \cos(0.8x)$; 8) $\cos(x/2)e^{-x^2+3x}$; 9) $\sin(0.6x)e^{-x^2}$

Функции $\Psi(X)$:

- 1) $e^{-2(2x-2)}$; 2) 0.01; 3) e^x ;
 4) e^{-2x} ; 5) e^{3x} ; 6) $1/\sqrt{2\pi}$;
 7) e^{2x-3} ; 8) e^{3x-2} ; 9) $e^{x-1.5}$;

3.2. $\Delta x_0 = 0.5$, $x_0 = -1$, $b = 2\pi - 1$, $y_0 = 8$, $g(x) = -\sin(x+1)$

Функции $\Phi(x)$:

- 10) $\sin(x+1)$; 11) $e^{-\cos(x+1)}$; 12) $(x+2)e^{-\cos(x+1)}$;

Функции $\Psi(x)$:

- 10) $1/6 \cos(x+1)$; 11) $1/8$; 12) $4\cos(x+1)$.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

1. Изучение методов Рунге-Кутта.
2. Составление блок-схемы алгоритма.
3. Составление программы расчёта (в программе предусмотреть возможность её использования для различных функций $f(x,y)$).
4. Ввод программы в ЭВМ и отладка программы.
5. Проведение вычислительных экспериментов.
6. Анализ полученных результатов.

6. УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ОТЧЕТА

Каждый студент оформляет отчёт, который должен содержать :

- постановку задачи;
- блок-схему алгоритма численного решения;
- распечатки отладок программы;
- распечатки результатов расчёта;
- анализ результатов;
- перечень ошибок, допущенных в процессе прохождения задания;
- ответы на контрольные вопросы.

В ответе следует дать необходимые пояснения использованных в программе имён переменных .

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем объясняется различие точности методов Рунге-Кутта 1, 2, 3, 4-го порядка?
2. Каким образом в программе обеспечивается выход на точку $X = b$?
3. Какова реальная максимальная точность численного метода решения дифференциального уравнения на конкретной ЭВМ ?

ЛАБОРАТОРНАЯ РАБОТА № 10

РАСЧЕТ ТРАЕКТОРИИ ПОЛЕТА

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является:

- овладение практическими навыками численного решения дифференциальных уравнений второго порядка и систем дифференциальных уравнений;
- практика и использование стандартной программы решения систем обыкновенных дифференциальных уравнений;
- проведение вычислительных экспериментов по решению практических задач.

2. ОСНОВЫ ТЕОРИИ

При интегрировании дифференциальных уравнений высших порядков можно использовать два подхода. Во-первых, дифференциальное уравнение может быть непосредственно преобразовано в конечно-разностную форму путем представления всех его производных соответствующими конечно-разностными выражениями. Например, уравнение падения тела

$$\frac{d^2Y}{dt^2} = -G(Y), \quad Y(t=0) = h, \quad V(t=0) = V_n,$$

где Y - высота, t - время, $G(Y)$ - ускорение падения, h, V_n - начальная высота и скорость, может быть численно решено с помощью конечно-разностного уравнения

$$Y_{i+1} = 2Y_i - Y_{i-1} + (\Delta t)^2 \cdot G(Y_i),$$

в котором точки Y_0 и Y_1 находятся из начальных условий:

$$Y_0 = h,$$

$$Y_1 = Y_0 + V_n \cdot \Delta t$$

Во-вторых, дифференциальное уравнение высшего порядка может быть представлено в форме системы дифференциальных уравнений первого порядка, которые решаются последовательно рассмотренными ранее методами. Например, написанное выше уравнение падения тела может быть представлено системой

$$\begin{cases} \frac{dY}{dt} = V(t), \\ \frac{dV}{dt} = -G(Y) \end{cases}$$

Решение данной системы методом Эйлера дает систему конечно-разностных уравнений

$$V_{i+1} = V_i + \Delta t(-G((Y_i))),$$

$$Y_{i+1} = Y_i + \Delta t \cdot \left(\frac{V_i + V_{i+1}}{2} \right),$$

где $V_0 = V_n, Y_0 = h$. Аналогично может быть применен метод Рунге-Кутта.

3. ОБЪЕКТ ИССЛЕДОВАНИЯ, ОБОРУДОВАНИЕ

Объектом исследования являются элементы численных методов и алгоритмического языка C++.

Для выполнения лабораторной работы требуется наличие электронной вычислительной техники с установленным программным обеспечением.

4. ВАРИАНТЫ ЗАДАНИЙ

Дана система дифференциальных уравнений второго порядка, описывающих траекторию движения тела переменной массы в воздухе,

$$\begin{cases} m(t) \frac{d^2 X}{dt^2} = [P(t) - R(V)] \cos[\alpha(t)]; \\ m(t) \frac{d^2 Y}{dt^2} = [P(t) - R(V)] \sin[\alpha(t)] - m(t) \cdot g(Y); \\ X = X_0, Y = Y_0, V = V_0, \alpha = \alpha_0, t = 0. \end{cases}$$

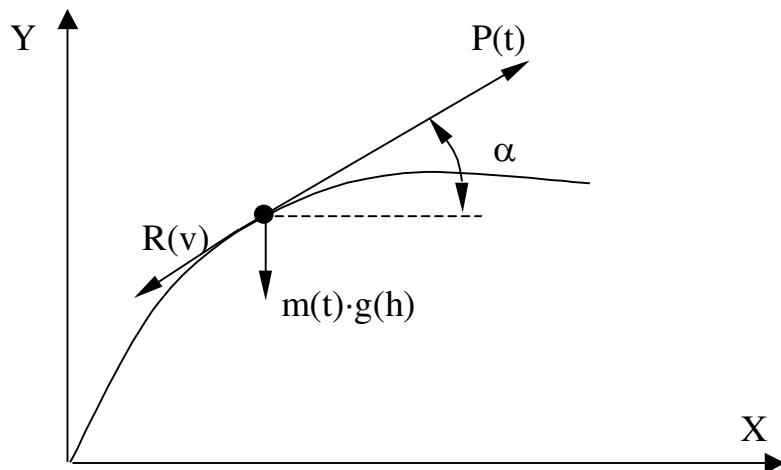


Рис. 3. Схема действующих сил

В приведенной системе уравнений X, Y - пространственные координаты; t - время; α - угол, образуемый касательной к траектории движения с осью X ; m - масса тела, изменяющаяся по заданному закону;

$R = (C_x \rho V^2 S_m) / 2$ - сила сопротивления воздуха движению тела (C_x - константа, ρ - плотность воздуха, изменяющаяся с высотой движения Y ; S_m - площадь сечения тела); g - ускорение свободного падения; V - полная скорость движения; P - сила, создаваемая истекающим потоком массы, изменяется по заданному закону.

Каждому студенту предлагаются по заданию преподавателя определенные начальные условия и закономерности $m(t)$, $P(t)$, $\rho(Y)$.

Необходимо численно решить данную задачу двумя способами;

1) непосредственно методом конечных разностей получить решение задачи, используя конечно-разностное представление второй производной;

2) привести дифференциальные уравнения второго порядка к системе дифференциальных уравнений первого порядка, численно решить эту систему с помощью стандартной подпрограммы.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И МЕТОДИЧЕСКИЕ рекомендации .

1. Вывести конечно-разностную формулу численного интегрирования.
2. Разработать алгоритм решения задачи.
3. Первые десять шагов интегрирования произвести вручную.
4. Составить программы решения задачи по разработанному алгоритму и с использованием стандартной подпрограммы.
5. Проанализировать результаты расчета.

6. УКАЗАНИЯ ПО ОФОРМЛЕНИЮ ОТЧЕТА

Каждый студент оформляет отчет, который должен содержать:

- постановку задачи;
- разностную формулу решения;
- блок-схему алгоритма;
- таблицу результатов ручного расчета;
- распечатки программ и отладок программ расчета;
- распечатки результатов расчетов;
- анализ результатов (проиллюстрировать графически);
- ответы на контрольные вопросы.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким образом можно численно решить данную задачу, если часть граничных условий будет задана не в начальной точке (например, в конечной точке траектории или в другой точке) ?

2. Какую минимальную величину допустимой погрешности решения можно задавать?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК**Основная литература**

1. Острайковский, В. А. Информатика : учебник для вузов / В. А. Острайковский .— 5-е изд., стер. — М. : Высш. Шк., 2009 .— 512 с.

Дополнительная литература

1. Информатика. Базовый курс: учеб. пособие для вузов / под ред. С. В. Симоновича. СПб. [и др.] : Питер, 2000. 640 с. : ил.
2. Информатика: Базовый курс: Учеб.пособие для вузов / Под ред.С.В.Симоновича. 2-е изд. М.[и др.] : Питер, 2005. 640с. : ил.
3. Павловская, Т.А. С/C++: Структурное программирование: Практикум / Т.А.Павловская, Ю.А.Щупак. СПб.и др. : Питер, 2004. 240с.
4. Павловская, Т.А. С/C++: Программирование на языке высокого уровня. Структурное программирование : Учеб.пособие для вузов / Т.А.Павловская, Ю.А.Щупак. СПб.и др. : Питер, 2002. 240с.