Министерство образования Республики Беларусь

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«Брестский государственный технический университет»

Кафедра информатики и прикладной математики

Методические рекомендации

по выполнению курсовых работ по дисциплине **«Информатика»** для специальностей 36 01 01 «Технология машиностроения», 36 01 03 «Технологическое оборудование машиностроительного производства», 27 01 06 «Техническая эксплуатация автомобилей»

Издание третье, дополненное

Брест 2005

УДК 681.3

Методические рекомендации содержат сведения о требованиях к содержанию и оформлению курсовых работ, структуре программы, базовых структурах программирования, примеры решения некоторых типовых задач и стандартные процедуры. Примеры приведены на языке QBASIC.

Предназначены для студентов первого и второго курсов специальностей "Технология машиностроения", "Технологическое оборудование машиностроительного производства", "Техническая эксплуатация автомобиля" по дисциплинам "Информатика" очной и заочной форм обучения и имеют целью оказать помощь студентам в подготовке и оформлении курсовых работ по названным дисциплинам.

Издание второе, дополненное.

Составитель: В. Л. Быков, доцент, к. т. н.

Рецензент: В. М. Мадорский, заведующий кафедрой информатики и прикладной математики Брестского госуниверситета, доцент, к. ф. м. н.

Содержание

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Выбор варианта задания

Вариант выполнения курсовой работы выбирается по первому символу фамилии студента и двум последним номерам зачетной книжки. Если номер зачетной книжки больше числа вариантов, то берется вариант по последнему номеру зачетной книжки. Студенты, фамилии которых начинаются на буквы:

А, Б, В, К, Л, М, У, Ф, Щ выбирают варианты заданий из приложения 1;

Г, Д, Е, Н, О, П, Х, Ц , Э выбирают варианты заданий из приложения 2;

Ж, З, И, Р, С, Т, Ч, Ш, Ю, Я выбирают варианты заданий из приложения 3.

Например, если у Студента Жук В. Д. номер зачетной книжки 280305, тогда номер задания – Приложение 3, Рис. 3.5; если у студента Мамонова И. Б. номер зачетной книжки 289025, тогда номер задания - Приложение 1, Рис. 1.25; если у студента Николаева С. Т. номер зачетной книжки 289332, тогда номер задания будет выбираться из Приложение 2, Рис.2.

Порядок выбора варианта может быть изменен по указанию преподавателя.

1.2. Требования к содержанию курсовой работы

Курсовая работа выполняется студентом самостоятельно с использованием языка программирования Бейсик (Паскаль и др.) или с использованием электронной таблицы Excel.

Объем и содержание курсовой работы должны быть достаточными для проверки знаний студента по изучаемой дисциплине в объеме курса. Работа может быть расчетной, графической или смешанной, расчетно-графической. Общий объем работы, в зависимости от содержания, должен составлять 15 - 20 листов машинописного текста. Программа должна быть выполнена с использованием принципов структурного программирования.

Задание на выполнение курсовой работы выдается преподавателем или выбирается студентом самостоятельно по одной из профилирующих кафедр и согласовывается с преподавателем. Допускается в качестве темы курсовой работы выбирать выполненные или выполняемые курсовые работы по другим дисциплинам.

Варианты типовых тем курсовых работ

А. Расчетные:

разработать программу для ввода блока данных, сохранения их на диске, загрузки, обработки и вывода результатов вычислений на экран и печать в виде отформатированных таблиц.

Б. Расчетно-графические:

разработать программу для ввода блока данных, сохранения их на диске, загрузки, обработки данных и вывода результатов на экран и печать в виде графиков функций и диаграмм;

разработать демонстрационную программу для решения линейных и трансцендентных уравнений одним или двумя способами, анализа функций одной и двух переменных.

В. Графические:

разработать программу для моделирования изображения детали (аксонометрия и проекции). Программа должна обеспечивать изменение масштаба, поворот изображения, перемещение изображения в любую точку экрана с контролем выхода изображения за границы экрана;

разработать программу моделирования движения механизма. Программа должна обеспечивать изображение механизма в статике, динамике, построение зон действия звеньев механизма, построение траектории движения заданной точки механизма и др.

Содержание курсовой работы

Курсовая работа должна включать: титульный лист, бланк задания, содержание, введение (не обязательно), пояснительную записку, заключение (не обязательно), список использованной литературы, приложение.

Таблица 1.1

Описание переменных <- заголовок таблицы

Имя программы	Назначение программы		Лист	
			Листов	
Переменная			Комментарий	
обозначение в формуле	имя переменной	тип переменной		

Рис.1.1. Пример оформления таблицы

Пояснительная записка должна содержать наименование темы, цели работы, постановку задачи, исходные данные, краткие теоретические сведения об используемых методах оптимизации, рисунок механизма или детали в аксонометрии, расчетные формулы, описание переменных, схемы алгоритмов. В пояснительную записку целесообразно включать общую схему алгоритма и полную схему алгоритма одного из блоков программ по выбору студента.

Описание переменных представляется в виде таблицы (рис.1.1).

В приложение включаются распечатки паспорта программы, меню, алгоритма программы, таблиц результатов вычислений, графиков функций и диаграмм, иллюстрирующих возможности программы, перечень использованных операторов и команд без комментариев.

Распечатки программ не должны содержать никаких комментариев к тексту программы, кроме названий процедур.

1.3. Требования к оформлению курсовых работ

Пояснительная записка и приложения оформляется в соответствии с требованиями ГОСТ 2.105-79. ЕСКД: «Общие требования к текстовым документам».

Весь текст пояснительной записки пишется ***от руки*** на стандартных листах формата А4. Текст должен иметь поля: слева - 30 мм, справа - 10 мм, сверху - 15 мм, снизу - 20 мм. Средняя плотность текста по вертикали 28 - 30 строк на страницу. Титульный лист оформляется на ЭВМ с использованием одного из редакторов текста. Пример оформления титульного листа приведен на рис.1.2. Остальные приложения распечатываются на принтере.

Текст курсовой работы разделяется на разделы и подразделы. Разделы нумеруются арабскими цифрами с точкой, например, 1., 2. и так далее. Если в разделе есть подразделы, то они нумеруются в пределах данного раздела (1.1., 1.2. ...). Заголовки разделов и подразделов записываются заглавными буквами. Точка в конце заголовка не ставится. Заголовки отделяются от текста одной свободной строкой.

Все страницы курсовой работы нумеруются, на первой странице номер не ставится. Нумерация страниц должна быть сквозная, включая приложения.

Формулы, используемые в пояснительной записке, записываются в общем виде в отдельной строке и нумеруются. Номер формулы записывается арабскими цифрами у правого края листа и заключается в круглые скобки. Ниже формулы приводится пояснение имеющихся в ней символов. Описание каждого символа приводится в отдельной строке. Пояснения к символам, кроме последнего, заканчиваются точкой с запятой, а пояснения к последнему символу - точкой.

Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра вычислительной техники и прикладной математики

Тема: *Исследование функции одной переменной*

Выполнил: студент группы № _____

_____

(Фамилия, инициалы)

Консультант: _____

(Фамилия, инициалы)

Проверил: _____

(Фамилия, инициалы)

г. Брест, 2001 г.

Рис.1.2. Пример оформления титульного листа

Ссылки на формулы в тексте пояснительной записки указываются в виде их номеров в круглых скобках.

Таблицы, если их несколько, нумеруются в пределах раздела. Боковые и нижние ограничивающие линии не изображаются (рис.1.1,). Если таблица разделена на несколько частей, то номер таблицы и заголовок пишутся только над первой частью, а над последующими частями пишут " Продолжение таблицы" и ее номер. Ссылки на номера таблиц указываются следующим образом: Табл. 3.1.

Рисунки выполняются на отдельных листах. Все рисунки, также как и формулы, нумеруются в пределах раздела.

Различная ориентация текста, таблиц и рисунков не допускается.

В тексте пояснительной записки разрешается использовать без расшифровки только общепринятые сокращения. Если используются нестандартные сокращения, то они должны быть расшифрованы в тексте при первом упоминании. Использовать сокращения отдельных слов не разрешается.

Схемы алгоритмов выполняются в соответствии с требованиями ГОСТ 19.003-80. Основные элементы схем алгоритмов приведены в табл. 1.1. Соотношение между геометрическими элементами схем установлены стандартом: размер *a* выбирается из ряда 10, 15, 20 мм. Допускается увеличивать размер *a* на число кратное 5. Размер *b* равен 1,5*a*, допускается устанавливать размер *b* равным 2*a*.

Список использованных источников составляется в алфавитном порядке или в порядке ссылок на источники в тексте. Примеры правильной записи некоторых источников приведены в списке использованной литературы настоящих рекомендаций. Номер источника проставляется в конце фразы или данных, заимствованных из соответствующего источника, и заключается в квадратные скобки, например [5].

Приложение должно начинаться с новой страницы. Если приложений несколько, то каждое из них должно начинаться с новой страницы. В верхнем правом углу пишется заглавными буквами слово "ПРИЛОЖЕНИЕ" и его номер, а в центре следующей строки пишется заглавными буквами название приложения.

2. ОСНОВЫ ПРОГРАММИРОВАНИЯ

2.1. Этапы разработки программы

Всякая программа проходит от момента своего появления до списания определенный жизненный цикл. Основными этапами этого цикла являются: содержательная постановка задачи, математическая постановка задачи, выбор метода решения, разработка математической модели, разработка схемы алгоритма, написание программы на одном из языков программирования, отладка программы, сдача программы в эксплуатацию и научно-техническое сопровождение программы.

Содержательная постановка задачи заключается в формулировке задачи на естественном языке.

Математическая постановка задачи сводится к точному описанию исходных данных, условий задачи и целей ее решения с использованием математических выражений в общем виде.

Математическая модель задачи представляет собой совокупность математических выражений, описывающих данную задачу.

Метод решения задачи выбирается из известных методов. При отсутствии известных методов, разрабатывается свой, оригинальный метод решения. После выбора метода решения осуществляется *формализация* задачи или, иными словами, описание исходных данных и условий задачи в виде, удобном для ввода в ЭВМ; неформализованные условия задачи представляются в математической форме.

Схема алгоритма является одним из способов наглядного представления алгоритма с помощью специальных элементов, предусмотренных ЕСКД. Некоторые из этих элементов приведены в табл.1.1.

Алгоритм - точное, однозначное предписание последовательности действий (операций), приводящее к решению задач данного класса за конечное число шагов или заданное время.

Написание программы на одном из языков программирования заключается в описании схемы алгоритма с помощью операторов и команд соответствующего языка высокого уровня.

Отладка программы заключается в проверке правильности функционирования алгоритма решения задачи с помощью контрольных примеров, результаты решения которых заведомо известны, устранении обнаруженных синтаксических и логических ошибок.

Научно-техническое сопровождение программы предусматривает контроль за результатами работы программы и устранение ошибок, обнаруженных в процессе эксплуатации, доработки программы и ее совершенствования в соответствии с требованиями заказчика.

2.2. Структура программ

При разработке программы необходимо руководствоваться принципами структурного программирования. К ним относятся ***нисходящая разработка, модульное проектирование*** и ***использование базовых структур***.

При реализации принципа нисходящей разработки программа разрабатывается в следующей последовательности:

1. На основе анализа задача разбивается на подзадачи, выделяются уровни и подуровни, составляется иерархическая структура программы;

2. Для каждого уровня определяется:

- метод решения и разрабатывается математическая модель задачи;
- определяются основные блоки программы и разрабатывается укрупненная схема алгоритма;
- определяются входные и выходные переменные, общие для данного уровня;
- разрабатываются схемы алгоритмов для реализации основных блоков программы, определяются входные и выходные переменные соответствующего уровня.

Этот процесс продолжается, пока схема алгоритма не будет доведена до базовых структур соответствующего языка программирования. Базовые элементы схем алгоритмов для всех языков программирования в основном совпадают. Отличия могут заключаться только в форматах используемых операторов. Основные базовые элементы схем алгоритмов приведены в табл. 2.1.

Для каждой программы составляется описание переменных по форме, приведенной на рис.1.1. Это позволит избежать ошибок при программировании, например, повторного использования имен переменных; систематизировать исходные данные и результаты вычислений; облегчит составление отчетной документации.

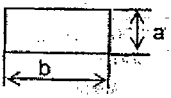
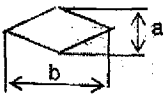
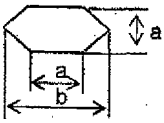
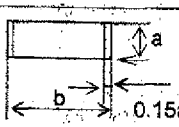
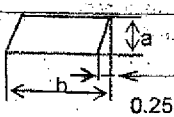
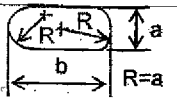
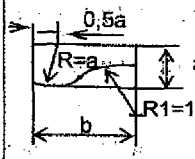
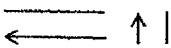
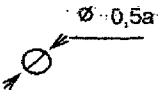
Принцип модульного проектирования заключается в следующем: при программировании различных задач всегда встречаются фрагменты программ, которые используются в теле программы неоднократно, например, вычисление факториала, вычисление производной, построение графика функции, вызов меню пользователя и так далее. В этих случаях такие типовые блоки программы могут быть оформлены в виде подпрограмм, процедур или функций пользователя и использоваться не только в разрабатываемой программе, но и в других программах.

Программа для курсовой работы должна включать в общем случае следующие программные модули (блоки): паспорт программы, блок ввода данных, меню программы, блок вычислений, блок вывода результатов на печать, блок построения графиков и диаграмм, блоки сохранения результатов вычислений на диске и загрузки результатов предыдущих вычислений с диска, блок выхода из программы. Выход из программы должен осуществляться только через меню программы, поэтому каждый программный модуль должен содержать процедуру возврата в меню.

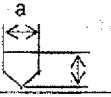
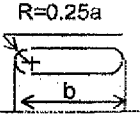
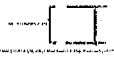
Паспорт программы содержит наименование программы, ее назначение и область применения, сведения об авторе и дате разработки или последней модификации, адрес и номера телефонов, факсов для связи с автором или фирмой - разработчиком программы, обладателем лицензии.

Таблица 1.1

Основные графические элементы схем алгоритмов

Наименование	Обозначение	Функция
Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды, или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы)
Предопределенный процесс		Использование созданных ранее или отдельно описанных алгоритмов или программ
Ввод – вывод		Преобразование данных в форму, пригодную для обработки(ввод) или отображения результатов обработки (вывод)
Дисплей		Ввод данных с подключенного к компьютеру дисплея или вывод данных на дисплей
Документ		Ввод-вывод данных, носителем которых служит бумага
Линии потока		Указание на последовательность связи между символами. Можно без стрелки, если линия направлена слева направо или сверху вниз, со стрелкой в остальных случаях
Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы в пределах листа

Продолжение таблицы 1.1

Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы на разных листа
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементами схемы с пояснениями

Блок ввода данных содержит объявления типов переменных, массивов, присвоение начальных значений переменным, элементам массивов, символьным константам.

Примечание: согласно ГОСТ 19.701 – 90 размеры элементов схем не регламентируются.

Меню программы содержит список разделов программы, к которым можно обращаться непосредственно после загрузки программы. В состав пунктов меню могут входить следующие разделы: ввод данных, расчеты, построение графиков функций, вывод результатов на печать или экран, сохранение результатов расчетов на диске, выход из программы. Каждый пункт меню может содержать вложенное меню, обеспечивающее дополнительный выбор вариантов использования программы. Такой подход непосредственно вытекает из принципов структурного программирования.

Блок вычислений служит для реализации вычислений в соответствии с разработанной математической моделью. Он может содержать также модули сохранения результатов расчетов в файле на диске, вывода предварительных и итоговых результатов на экран в табличной форме, возврата в главное меню.

Блоки вывода результатов и вывода результатов на печать должны содержать подменю и позволять выводить результаты вычислений на экран или на печатающее устройство в табличной форме и/или в виде графиков и диаграмм по желанию пользователя. Выводиться должны текущие результаты или результаты последних вычислений без предварительного расчета. Необходимо предусмотреть защиту от ошибочных ситуаций, например, отсутствия данных.

При профессиональном программировании принято весь экран делить на три зоны:

- первая зона - строки 1 - 5. В эти строки выводятся пункты горизонтального меню, сообщения и запросы программы;

- вторая зона - рабочая, расположена между первой и третьей зонами. В эту зону выводятся результаты вычислений, вертикальное меню, графики функций и тому подобное;

- третья зона - строки 22-24, в них выводятся подсказки программы, назначение клавиш управления (навигационное меню).

Указанные соображения необходимо учитывать в дальнейшем при размещении информации на экране.

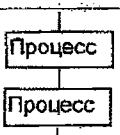
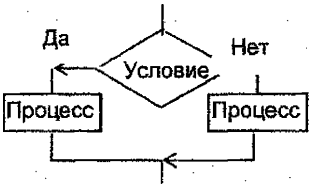
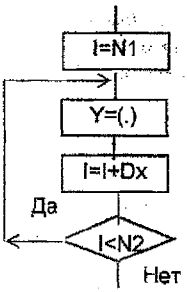
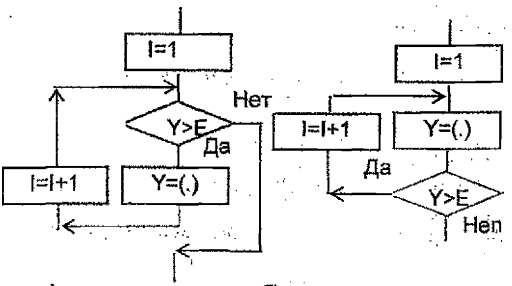
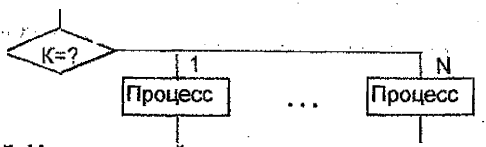
3. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1. Ввод данных

Для ввода данных используются операторы LET, INPUT, DATA, READ, LINE INPUT, функция INPUT$.

Таблица 2.1

Представление базовых структур алгоритмов

Схема алгоритма	Запись на алгоритмическом языке
А. Следование 	*Нач* <выражение> ... <выражение> *Кон*
Б. 	*Если* <условие> *то* <выражение> *иначе* <выражение> *Все*
В. Цикл с заданным числом повторений (цикл "ДО") 	*Для* I от N1 до N2 шаг Dx *нц* <выражение> ... <выражение> *кц* --- For i=N1 TO N2 Step Dx < Тело цикла> Next i
Г. Цикл с параметром или цикл типа "Пока"  а) с предусловием б) с постусловием	*Пока* <условие> *нц* <выражение> ... <выражение> *кц* --- While <условие> <Тело цикла> Wend
 Д. Множественный переход	*Выбор* при условии 1:<выраж.> ... при условии N:<выраж.> *Все*

Оператор **LET**. Служит для присвоения значений переменным. Формат оператора: LET <имя переменной> = <выражение>.

Оператор LET может быть опущен, поэтому выражения

LET A=exp(x)+ sin(x) и

A=exp(x)+ sin(x)

эквивалентны.

Оператор **INPUT** служит для ввода данных с клавиатуры в режиме диалога с пользователем. Формат оператора:

INPUT [;] "текстовое выражение"[;/,]<список переменных>

Здесь [;] – необязательный параметр, оставляет курсор в текущей строке; [;/,] - означает, что в качестве разделителя может использоваться один из указанных знаков: ";" или ",".

При выполнении оператора INPUT на экран выдается запрос на ввод данных. Если в качестве разделителя используется ";", то запрос сопровождается выводом на экран вопросительного знака, при использовании в качестве разделителя "," вопросительный знак на экран не выводится. Текстовое сообщение позволяет сделать запрос понятным пользователю. В списке переменных данные разделяются запятыми или пробелами (предпочтительнее разделять данные запятыми). С помощью оператора INPUT можно вводить как числовые, так и символьные переменные. Если символьная переменная не содержит пробелов и других разделителей, то при вводе данных заключать ее в кавычки не обязательно.

Примеры 3.1.1:

INPUT "Введите переменные A и B"; A,B

INPUT "Введите Ваш год рождения", GR$

Здесь A и B - числовые переменные, GR$ - символьная переменная.

Оператор **LINE INPUT** служит для ввода одной символьной переменной. При вводе значение символьной переменной в кавычки не заключается, в ней допускается наличие пробелов и других разделителей. Конец строки определяется символом возврата каретки - нажатие клавиши ENTER. Формат оператора LINE INPUT аналогичен формату оператора INPUT.

Функция **INPUT$** служит для ввода символов, не отображаемых на экране, например, пароля. Формат функции: INPUT$(n [,[#]nf])

Здесь n - число вводимых символов, # - номер канала при вводе данных из файла, nf - имя файла.

Примеры 3.1.2:

LINE INPUT "Введите фамилию, имя и отчество ",FIO$

B$=INPUT$(5) - символьной переменной B$ присваивается код из пяти символов, при выдаче запроса вводимые символы отображаться на экране не будут. Когда в программе встречается оператор INPUT$(n), программа останавливается и ожидает ввода данных. Поэтому данный оператор в формате B$= Input$(1) может быть использован для остановки программы до нажатия любой клавиши.

Операторы **DATA** и **READ** позволяют вводить данные из программы. Оператор DATA заносит данные в специальную область оперативной памяти компьютера, а оператор READ считывает эти данные из оперативной памяти и присваивает их переменным. Форматы операторов:

DATA <список данных>

READ <список переменных>

Пример 3.1.3:

DATA 125, 34.78, 1.24E-5, БРЕСТ, "МИНСК - СТОЛИЦА"

READ A,B,D,C$,C1$

Переменным A, B и D будут присвоены, соответственно, числовые значения 125, 34.78 и 0.0000124, переменным C$ и C1$ - "БРЕСТ" и "МИНСК - СТОЛИЦА". Если число переменных в операторе READ больше числа значений в операторе DATA, то будет выдано сообщение об ошибке.

Для повторного использования данных используется оператор RESTORE. Формат оператора: RESTORE [<метка>].

Пример 3.1.4: **Использование операторов DATA, READ, RESTORE.**

```
10 REM ПРИМЕР ИСПОЛЬЗОВАНИЯ ОПЕРАТОРОВ DATA, DEAD, RESTORE
20 DATA 136.75, 18E5,123.45,1978,.9875
25 DATA .5439,1.567,4.65,12.23,48.56
30 READ A1,A2,A3,A4,a5 : REM чтение данных из строки 20

150 READ B1,B2 : REM чтение данных из строки 25
...
250 RESTORE 25 : REM перевод указателя данных на метку 25
260 INPUT "Введите размерность массива N, не более 5",N
260 DIM D2(N)
270 FOR i=1 TO N
280 READ D2(i) : REM чтение данных из строки 25 в одномерный массив
290 NEXT i
```

3.2. Условный оператор IF

Различают однострочные и многострочные условные операторы.

Форматы однострочного оператора IF:

а) простой однострочный оператор

IF <условие> THEN <операторы>

При выполнении оператора IF проверяется условие и, если оно истинно, то выполняется действие, указанное после оператора THEN. Если выражение ложно, то управление передается на оператор, следующий за оператором IF.

Примеры:

```
IF x<e THEN GOTO m1
IF x<a THEN y=SIN(x)*EXP(2^LOG(x)): GOTO m2
```

б) простой расширенный оператор IF

IF <условие> THEN <операторы1> ELSE <операторы2>

При выполнении оператора IF, если условие истинно, то выполняются операторы, указанные после оператора THEN, в ином случае выполняются операторы, следующие за оператором ELSE. После выполнения соответствующей группы операторов управление передается на оператор, следующий за оператором IF.

После операторов THEN и ELSE может быть указано несколько операторов, разделенных двоеточием. Однако число операторов ограничено длиной строки.

Форматы многострочного оператора IF:

а) простой	б) расширенный

IF <условие> THEN
<первая группа операторов>
ELSE
<вторая группа операторов>
END IF
	IF <условие> THEN
	<первая группа операторов>
	ELSEIF <условие> THEN
	<вторая группа операторов>
	ELSE
	<третья группа операторов>
	END IF

При записи операторов следует обращать внимание на структуру записи. Структура должна соответствовать той, что указана в примере.

Достоинство многострочного IF состоит в том, что число операторов в группах не ограничено.

3.3. Операторы цикла

Циклом называется процедура, в которой вычислительные операции выполняются многократно заданное число раз или до достижения некоторой переменной, вычисляемой в теле цикла определенного, наперед заданного значения. Циклы первого типа называются циклами типа "ДО", а циклы второго типа - циклами типа "ПОКА".

Циклы бывают двух типов: циклами с предусловием и циклами с постусловием. В циклах с предусловием вначале проверяется условие окончания цикла и, если условие окончания цикла не выполняется, то выполняется тело цикла. В таких циклах значение переменной, проверяемой в теле цикла, должно быть вычислено заранее или ей должно быть присвоено некоторое значение, заведомо большее условия окончания цикла, до входа в цикл. В циклах с постусловием условие окончания цикла проверяется в конце цикла.

Циклы могут быть организованы с использованием оператора IF или с использованием специальных операторов: FOR/NEXT, WHILE/WEND, DO/LOOP.

Оператор FOR служит для организации циклов с заданным числом повторений. Цикл относится к циклам с постусловием. Формат оператора:

```
FOR i=Iнач TO Iкон STEP di
  <тело цикла>
Next i
```

В данном формате Iнач – начальное значение переменной цикла, Iкон – конечное значение переменной цикла, а di – шаг приращения значения аргумента.

Операторы FOR и NEXT образуют операторные скобки, между которыми заключено тело цикла.

Оператор WHILE служит для организации циклов с заданным числом повторений, относится к циклам с постусловием. Условием окончания цикла является достижение функцией, вычисляемой в теле цикла, заданного значения. Формат оператора:

```
WHILE <условие>
                <тело цикла>
WEND
```

Тело цикла выполняется в том случае, когда условие истинно. Если условие ложно, то программа выходит из цикла. Особенностью использования данного цикла является то, что значение вычисляемой функции должно быть известно перед входом в цикл. Иначе, если начальное значение функции меньше заданной величины ε, программа никогда не войдет в цикл.

Оператор DO - универсальный оператор служит для организации циклов типа "ПОКА". Он может быть организован как цикл с предусловием, так и как цикл с постусловием. Когда в теле цикла используется оператор WHILE, то тело цикла выполняется в том случае, если условие истинно. Когда используется оператор UNTIL, то тело цикла выполняется в том случае, если условие ложно.

Оператор DO

а) цикл с предусловием

```
DO <WHILE|UNTIL>
  <тело цикла>
LOOP
```

б) цикл с постусловием

```
DO
  <тело цикла>
LOOP <WHILE|UNTIL>
```

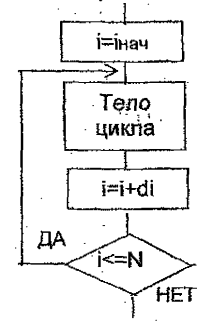
Примеры 4.3.1: **Организация циклов.**

а) цикл с постусловием

```
REM использование для организации цикла
REM оператора IF
M1:
i=iнач:    REM заголовок цикла
<тело цикла, вычисление функции F(i)>
i=i+di
IF i<=iкон THEN GOTO M1: REM конец цикла
PRINT "Результат"; F

REM использование оператора FOR
FOR i=iнач TO N STEP di
  <тело цикла, вычисление функции F(i)>
NEXT i
PRINT "Результат";F
```

Схема алгоритма

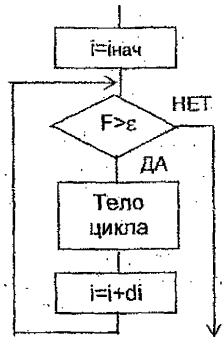


б) цикл с предусловием

```
REM Использование оператора IF
F=<выражение>
i=iнач
M1:
IF F<=ε THEN GOTO M2
  <тело цикла, вычисление F(i)>
  i=i+di
GOTO M1
M2:
PRINT "Результат"; F

REM Использование оператора
REM WHILE/WEND
F=<выражение>
i=iнач
WHILE F>ε
  <тело цикла, вычисление F(i)>
  i=i+di
WEND
PRINT "Результат";F
```

Схема алгоритма



3.4. Графические операторы

Оператр SCREEN устанавливает режим экрана. Формат оператора:

SCREEN режим [,[перекл_цвета] [,[актив_стр] [,видим_стр]]]

Здесь режим - целочисленная константа, устанавливает режим экрана. Может принимать значения от 0 до 13: 0 – текстовый режим, 16 цветов; 1 – графический режим, разрешающая способность 320X200, 4 цвета; 2- графический режим, разрешающая способность 640X200, 2 цвета; 9 – графический режим, разрешающая способность 640X350, 16 цветов; 12 – графический режим, разрешающая способность 640X480, 16, цветов; 13 - графический режим, разрешающая способность 320X200, 256 цветов;

перекл_цвета - целочисленная константа, определяет цветность экрана. Если зна-

чение параметра равно 0, то монохромное изображение. Если значение параметра больше единицы, то цветное изображение;

актив_стр - страница экрана, в которую записывается вывод текста или графики;

видим_стр - страница экрана, отображаемая на экране в данный момент.

При использовании оператора SCREEN ориентация осей координат соответствует 4-й координатной плоскости, то есть ось Y направлена вниз.

Пример 3.4.1:

```
SCREEN 9
LINE (110, 70)-(190, 120), , BF
LINE (0, 0)-(320, 200), 3, , &HFF00
```

Оператор VIEW определяет размер и положение области просмотра, где графика может быть выведена на экран, можно использовать для создания на экране нескольких окон просмотра. Формат оператора:

VIEW [[SCREEN] (x1,y1)-(x2,y2) [,[цвет] [,граница]]],

где опция SCREEN указывает, что координаты задаются относительно экрана, а не области просмотра;.

(x1,y1)-(x2,y2) - координаты диагонали противоположных углов области просмотра;

цвет - атрибут цвета, устанавливающий заполняющий цвет области просмотра;

граница - атрибут цвета, устанавливающий цвет границы области просмотра (бордюра).

Если все аргументы опущены, область просмотра - весь экран.

Допустимые атрибуты цвета зависят от используемого графического адаптера и режима экрана, установленного последним оператором SCREEN.

Пример 3.4.2:

```
SCREEN 9
VIEW (10, 10)-(300, 180), , 1
LOCATE 1, 11: PRINT " Первая область просмотра графики";
VIEW SCREEN (80, 80)-(200, 125),3, 5
LOCATE 11, 11: PRINT "Вторая область просмотра графики";
```

Оператор WINDOW переопределяет экранные координаты на математические. Формат оператора:

WINDOW [[SCREEN] (x1,y1)-(x2,y2)]

Данный оператор удобно использовать для установки требуемого масштаба изображения. Оператор разворачивает ориентацию оси Y. Ось Y направлена вверх.

Опция SCREEN инвертирует обычное направление декартовых координат так, что ось Y направлена на экране вниз;

(x1,y1) - логические координаты, соответствующие координатам верхнего левого угла области просмотра;

(x2,y2) - логические координаты, соответствующие координатам нижнего правого угла области просмотра.

WINDOW без аргументов выключает логическую систему координат.

Пример 3.4.3:

```
SCREEN 12
VIEW (10, 10)-(300, 180),1 , 4
WINDOW (-160, -100)-(160, 100)
REM построение окружности в центре экрана радиусом 100
CIRCLE (0, 0), 100
```

Операторы PSET и PRESET рисуют точку на экране.

```
PSET [STEP] (x,y) [,color]
PRESET [STEP] (x,y) [,color]
```

Опция STEP указывает, что x и y заданы относительно текущего графического положения курсора;

(x,y) - координаты точки устанавливаемой на экране;

color - атрибут цвета, устанавливаемый для точки. Если color опущен, то оператор PSET использует текущий цвет переднего плана, а PRESET использует текущий цвет фона (то есть точка не отображается на экране).

Доступные атрибуты цвета зависят от используемого графического видеоадаптера и режима экрана. Значения координат зависят от графического видеоадаптера, режима экрана и последних установок в операторах VIEW и WINDOW. При совместном использовании оператор PSET вычерчивает точку на экране, а оператор PRESET стирает точку.

Пример 3.4.4:

```
SCREEN 9
FOR i = 0 TO 320
   PSET (i, 100)
   FOR delay = 1 TO 100: NEXT delay
   PRESET (i, 100)
NEXT i
```

Оператор LINE рисует на экране линию или прямоугольник. Формат оператора:

LINE [[STEP](x1,y1)]-[STEP](x2,y2) [,[цвет] [,[B | BF] [,стиль]]]

Опция STEP - указывает, что координаты задаются относительно текущего графического положения курсора;

(x1,y1), (x2,y2) - координаты начала и конца линии на экране;

цвет - атрибут цвета, устанавливающий цвет линии или прямоугольника. Допустимые атрибуты цвета зависят от графического адаптера и режима экрана, установленного последним оператором SCREEN;

B - рисует прямоугольник вместо линии;

BF - рисует заполненный прямоугольник;

стиль - устанавливает стиль линии, представляет собой четырехразрядный шестнадцатеричный код. Если перевести шестнадцатеричный код в двоичный, то биты этого кода определяют, будут ли рисоваться точки на экране. Используется для изображения прерывистых и пунктирных линий. Например: 1110 1110 1110 1110 - двоичный код пунктирной линии. Код той же линии в шестнадцатеричном коде будет иметь вид &HEEEE; 1111 1111 1100 1100 – двоичный код штрих пунктирной линии. Код той же линии в шестнадцатеричном коде будет иметь вид &HFFCC. Символы H и & перед кодом числа являются признаком шестнадцатеричного числа.

Оператор CIRCLE рисует на экране окружности и эллипсы. Формат оператора:

CIRCLE [STEP] (x,y) радиус[,[цвет] [,[старт] [,[конец] [,сжатие]]]]

Опции

STEP - указывает, что координаты задаются по отношению к текущей графической позиции курсора;

(x,y) - координаты центра окружности или эллипса;

радиус - радиус окружности или эллипса в единицах текущей системы координат, определённой последними операторами SCREEN, VIEW и WINDOW;

цвет - атрибут цвета, устанавливающий цвет окружности;

старт - начальный угол дуги в радианах;

конец - конечный угол дуги в радианах. Знак "-" перед начальным или/и конечным значением угла позволяет соединить начало или конец дуги с центром окружности;

сжатие - отношение длины оси Y к длине оси X, используемое при изображении эллипсов.

Для перевода градусов в радианы используется формула:

(угол_в_градусах) * PI / 180

Пример 3.4.5:

```
SCREEN 12
PI=4*ATN(1) :  'Константа, вычисленная через арктангенс
REM окружность радиусом 200
CIRCLE (320, 100), 200
REM сектор эллипса, начальный угол равен 30 градусам,
REM конечный угол равен 120 градусам
CIRCLE STEP (200,100), 100,5,-30/180*PI,-120/180*PI,0.5
```

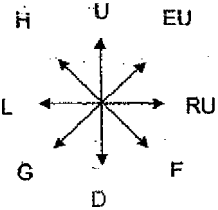
Оператор DRAW служит для вычерчивания фигур. Формат оператора:

DRAW строка_команд$,

где строка_команд$ - строковое выражение, содержащее одну или несколько команд оператора DRAW.

Команды оператора DRAW можно разделить на две группы: команды построения изображением и команды управления изображением.

Команды построения.



D[n] - перемещает курсор вниз на n единиц.
E[n] - перемещает курсор вправо-вверх на n единиц.
F[n] - перемещает курсор вправо-вниз на n единиц.
G[n] - перемещает курсор влево-вниз на n единиц.
H[n] - перемещает курсор влево-вверх на n единиц.
L[n] - перемещает курсор влево на n единиц.
R[n] - перемещает курсор вправо на n единиц.
U[n] - перемещает курсор вверх на n единиц.

[B] - префикс. Необязательная приставка, перемещение курсора без вычерчивания линии.

[N] - префикс. Необязательная приставка, нарисовать линию и возвратить курсор в первоначальную позицию.

M[{+|-}]x,y - перемещает курсор в точку x, y. Если перед x стоит + или -, то перемещает относительно текущей точки.

Команды управления.

Команды управления должны всегда предшествовать командам построения.

An - поворачивает объект на n * 90 градусов (n может быть 0, 1, 2 или 3).

Cn - устанавливает рисующий цвет (n- атрибут цвета).

Pn1,n2 - устанавливает цвет заполнения и границы объекта (n1 - атрибут цвета заполнения, n2 - атрибут цвета границы).

Sn - определяет масштаб рисунка, устанавливая единицу длины перемещения курсора. По умолчанию n равно 4, что эквивалентно 1 точке растра.

TAn - поворачивает угол на n градусов (от -360 до 360).

Если в командах изображения линии и перемещения курсора опущен параметр n, то курсор перемещается на 1 единицу.

Для выполнения подстроки команд DRAW из строки команд DRAW, используйте команду "X":

```
DRAW "X"+ VARPTR$(строка_команд$)
```

Пример 3.4.6:

```
SCREEN 9
Triangle$ = "F60 L120 E60"
DRAW "C2 X" + VARPTR$(Triangle$)
DRAW "BD30 P1, 2 C3 M-30,-30"
```

3.5. Внутренние и внешние функции

Если некоторую функцию необходимо вычислять в программе многократно, то целесообразно определить ее как функцию пользователя и использовать в дальнейшем так же, как и стандартную функцию языка программирования. Для этого язык QBASIC предоставляет возможность использовать внутреннюю функцию или внешнюю функцию. Внутренняя функция может использоваться только в текущей программе, а внешняя функция может использоваться как в текущей, так и в любой другой программе.

Внутренняя функция объявляется оператором DEF. При этом может быть два варианта использования данной функции: однострочная и многострочная.

Формат однострочной функции DEF:

DEF FNnn...nt(x_1,x_2, ...,x_m)=<выражение>

Здесь DEF – оператор; FN –стандартное имя функции; nn...n –расширение имени функции; t – тип переменной (результата вычисления функции); x_i – формальные параметры. Имя функции может содержать до 40 латинских символов. Число формальных параметров может быть шестнадцать.

Пример 3.5.1. Вычислить путь, пройденный автомобилем, если известны начальный путь ***S0***, скорость ***v***, ускорение – ***a***, и время движения автомобиля –***t***.

```
DEF FNput(S0,v,a,t)=S0+v*t+a*t^2/2
```

Формат многострочной функции DEF:

```
DEF FNnn...nt(x1,x2, ...,xm)
   y=<выражение>
   FNnn...nt=y
END DEF
```

Пример 3.5.2. Вычислить факториал числа N

```
DEF FNfactaria#(N)
   F=1
   FOR Fact =1 TO N
      F=F*Fact
   NEXT Fact
   FNfactaria#=F
END DEF
```

Используются функции пользователя так же, как и стандартные функции языка Бейсик. Например, вычислить факториал числа 15.

```
REM вычисление факториала

Input "Введите M ",M
y= FNfactaria#(M): PRINT "Факториал ";M;"равен"; y
```

Функция пользователя должна быть объявлена в начале программы, до ее использования.

Внешние функции описываются следующим образом:

FUNCTION <имя> (формальные параметры)

<тело программы>

<имя>=<результат>

END FUNCTION

Формат внешней функции аналогичен формату внутренней многострочной функции. Имя функции может начинаться с любых символов.

Используется внешняя функция так же, как и внутренняя. Однако при использовании внешней функции есть особенности. Одна из них состоит в том, что функция должна быть объявлена в вызывающей программе:

DECLARE FUNCTION <имя>(аргументы)

Другая особенность связана с использованием глобальных переменных. В данном пособии этот вопрос не рассматривается.

Подпрограммы

При необходимости многократного использования одних и тех же вычислительных процедур они могут быть оформлены в виде подпрограмм. Подпрограммы, так же как и функции, могут быть внутренними и внешними.

Внутренние подпрограммы должны быть согласованы по используемым переменным с главной программой. Вызов внутренних подпрограмм может быть организован с помощью операторов GOTO и GOSUB.

Примеры организации вызова подпрограмм с использованием операторов GOTO и GOSUB приведены на рисунке 3.6.1.

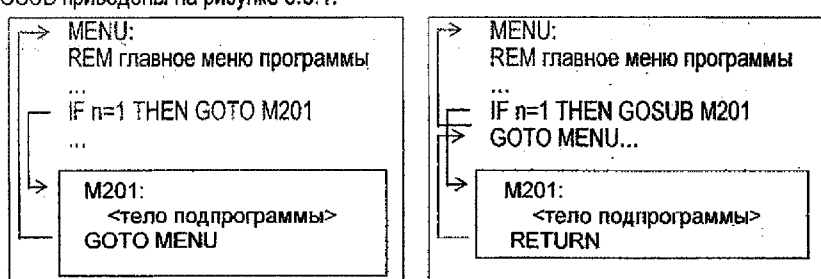


Рис.3.6.1. Организация вызова подпрограмм

Оператор **GOTO M** обеспечивает безусловный переход к указанной метке. В качестве метки может использоваться любой набор букв и цифр, заканчивающийся двоеточием, например, **MENU:**

Оператор **GOSUB M** вызывает подпрограмму по указанной метке. Подпрограмма завершается оператором **RETURN.** После выполнения подпрограммы управление передается на оператор, следующий за оператором, вызвавшим подпрограмму.

Внимание:

1. Если в подпрограмму вошли по оператору GOSUB, то выход из подпрограммы происходит по оператору RETURN на метку, следующую за оператором, вызвавшим подпрограмму. Из подпрограммы можно выйти в этом случае и с помощью оператора GOTO на любую метку.

2. Если в подпрограмму вошли по оператору GOTO, то выйти из подпрограммы можно только по оператору GOTO. Если в этом случае программа выходит на оператор RETURN, то программа ЗАВИСАЕТ.

Внешние подпрограммы оформляются с помощью оператора SUB.

Формат оператора:

```
SUB <имя> [список формальных параметров]
   <тело программы>
END SUB
```

По формату внешняя подпрограмма ничем не отличается от внешних функций. В качестве формальных параметров могут использоваться простые переменные и имена массивов любой размерности. Для их выделения в списке формальных параметров вслед за идентификатором массива ставятся круглые скобки, например:

```
SUB <имя_подпрограммы >(V1() As type1), V2 () AS type2) STATIC
```

В данном примере для массивов V1 и V2 явно указан тип массивов и указан способ распределения памяти под массивы – статический.

Вызов подпрограммы осуществляется оператором CALL.

Подпрограмма должна быть описана в вызывающей программе:

```
DECLARE SUB <имя_подпрограммы>(список параметров)
```

В подпрограмме не принято описывать массивы, указанные в числе фактических параметров, поэтому, чтобы узнать размерности массивов, применяют функции LBOUND(V,k) и UBOUND(V,k). Эти функции выдают, соответственно, значения нижней и верхней границ индекса массивов по заданному измерению k в массиве с именем V.

Пример 3.6.1. Подпрограмма суммирования элементов двухмерного массива:

```
SUB SummArray(A() As Real, S As Real)
   N1=LBOUND(A,1): M1=UBOUND(A,1)
   N2=LBOUND(A,2): M2=UBOUND(A,2)
   S=0
   FOR I=N1 TO M1
      FOR J=N2 TO M2
         S=S+A(I,J)
      NEXT J
   NEXT I
END SUB
```

Возврат из внешней подпрограммы всегда происходит на оператор, следующий за оператором CALL.

Файлы данных

Для хранения исходных данных и результатов вычислений могут использоваться файлы последовательного и файлы прямого доступа.

Файлы последовательного доступа.

Создание файла последовательного доступа:

```
OPEN "имя_файла" FOR OUTPUT AS #1
...
WRITE #1,<список значений>
CLOSE #1
```

Использование файла последовательного доступа

```
OPEN "имя_файла" FOR INPUT AS #1
...
INPUT #1,<список значений>
CLOSE #1
```

Файлы прямого доступа.

Открытие файла прямого доступа. Файл прямого доступа открывается одной командой как для чтения, так и для записи.

```
OPEN "имя_файла" FOR RANDOM AS #1
```

Описание буфера данных.

FIELD #1, N1 As V1$, N2 As V2$,...

где N1, N2 – длина поля, As – служебное слово, V1$, V2$ - имена полей буфера данных.

Создание файла прямого доступа:

а) запись данных в буфер:

LSET V1$ =<переменная символьного типа>

LSET V2$ =<переменная символьного типа>

...

б) запись данных из буфера в файл

PUT #1, nom

где nom – номер записи.

Использование файла прямого доступа:

а) чтение данных из файла в буфер данных:

GET #1, nom

б) чтение данных из буфера

<переменная>=V1$: <переменная>=V2$ ...

В файле прямого доступа могут храниться только символьные переменные. Поэтому числовые переменные при записи в буфер должны быть преобразованы в символьные эквиваленты с помощью следующих функций:

MKI$(N) – преобразует целое число одинарной длины в двухбайтовый символьный эквивалент;

MKL$(N) – преобразует целое число двойной длины в четырехбайтовый символьный эквивалент;

MKS$(N) – преобразует вещественное число одинарной точности в четырехбайтовый символьный эквивалент;

MKD$(N) – преобразует вещественное число двойной точности в восьмибайтовый символьный эквивалент.

Обратное преобразование переменных буфера в числовые переменные соответствующего типа осуществляется с помощью функций CVI(C), CVL(C), CVS(C), CVD(C).

4. ТИПОВЫЕ СХЕМЫ АЛГОРИТМОВ

4.1. Разработка паспорта и меню программы

Паспорт и меню программы представляют собой текстовую информацию. Для вывода ее на экран используются операторы PRINT, LOCATE, функции TAB(n), SPC(n). Для построения рамок проще всего использовать оператор LINE в графическом режиме.

Практический совет: Сначала напишите и отладьте программу в черно-белом цвете, а затем, при наличии свободного времени, занимайтесь украшением вашей программы всеми цветами радуги.

Пример 4.1.1 **: Паспорт программы**

```
REM Паспорт программы
CLS:
SCREEN 9
LINE(40,10)-(600,190),,b:
REM Для получения на экране и при печати линий равной
REM толщины проводится двойная вертикальная линия
LINE(41,10)-(601,190),,b:
LOCATE 3,20:?"Министерство образования Республики Беларусь"
LINE(100,25)-(540,25)
```

```
LOCATE 5,15:?" Брестский государственный технический университет"
LOCATE 6,12:?" Кафедра вычислительной техники и прикладной математики"
LOCATE 10,20:?"      КУРСОВАЯ РАБОТА"
LOCATE 11,20:?" тема: Моделирование изображения детали"
LOCATE13,12:?"Программа позволяет строить аксонометрическое изображение"
LOCATE 14,12:?"детали, и ее проекции, изменять масштаб изображения, пере-"
LOCATE 15,12:?"мещать деталь и поворачивать изображение на произвольный"
LOCATE 16,12:?"угол в режиме диалога с пользователем. Угол вводится в"
LOCATE 17,12:?"градусах"
LOCATE 19,25:?"Исполнитель: Баценко Д.Л. гр.Т-39"
LOCATE 20,25:?"Дата сдачи работы: 1.06.97"
LOCATE 23,35:?"Брест 1997"
```

Пример 4.1.2: **Меню программы**

```
MENU: REM Меню программы
DEFSTR c
REM переменные, имя которых начинается с символа "c" - символьные
SCREEN 2
LINE (115,20)-(515,140),,b : REM Построение двойной рамки
LINE (119,22)-(511,138),,b
LOCATE 5,22:?"*** Г Л А В Н О Е  М Е Н Ю ***"
LOCATE 10,22:?"1.Паспорт программы."
LOCATE 12,22:?"2.Проверка наличия корня."
LOCATE 14,22:?"3.Демонстрация процедуры вычисления."
LOCATE 16,22:?"4.Выход."
RETURN
```

4.2. Обработка пунктов меню

Для анализа номера выбранного пункта меню можно использовать три оператора: оператор IF, оператора ON GOTO или ON GOSUB, оператор SELECT CASE. Наиболее простой из перечисленных операторов - оператор ON GOTO/GOSUB. Наиболее удобные, с точки зрения структурного программирования , - многострочный оператор IF и оператор SELECT CASE.

Оператор IF. Для обработки пунктов меню можно использовать простой оператор IF:

IF <условие> THEN <оператор перехода>

IF <условие> THEN <оператор вызова подпрограммы>

Пример 4.2.1: **Обработка пунктов меню оператором IF**

```
OPM1: REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню

   1-й вариант                2-й вариант
IF nkl=1 THEN GOTO M1  |  IF nkl=1 THEN GOSUB M1: GOSUB MENU
IF nkl=2 THEN GOTO M2  |  IF nkl=2 THEN GOSUB M2: GOSUB MENU
. . .          . . .   |
IF nkl=n THEN GOTO Mn  |  IF nkl=n THEN GOSUB Mn: GOSUB MENU
LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1: REM Выдача звукового сигнала
SLEEP 3: REM Остановка на 3 секунды и очистка строки сообщения
LOCATE 1,1: PRINT "                                          "
GOTO OPM1: REM Возврат для ожидания нажатия клавиши
END
```

```
ONK: ' подпрограмма выбора пунктов меню (простейший вариант)
LOCATE 23,1: "Ваш выбор ": INPUT " ",nkl
RETURN
```

Здесь Mn - метка подпрограммы выполнения соответствующего пункта меню.

При использовании многострочного IF получается хорошо читаемая структура программы:

Пример 4.2.2: **Обработка пунктов меню многострочным IF**

```
OPM1: REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
    IF nkl=1 THEN
       GOSUB M11
       ...
       GOSUB MENU
    ELSEIF nkl=2
       GOSUB M21
       ...
       GOSUB MENU
       ...
    ELSE
      LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
      BEEP 1:  REM Выдача звукового сигнала
      SLEEP 3:  REM Остановка на 3 секунды и очистка строки сообщения
      LOCATE 1,1: PRINT "                                    "
      GOTO OPM1: REM Возврат для ожидания нажатия клавиши
    END IF
```

Пример 4.2.3 : **Обработка пунктов меню оператором ON GOTO/GOSUB**

```
OPM2: REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
```

Вариант 1	Вариант 2
ON nkl GOTO M1,M2,...,Mn	ON nkl GOSUB M1,M2,...,Mn

```
LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1:        REM Выдача звукового сигнала
SLEEP 3:       REM Останов на 3 секунды и очистка строки сообщения
LOCATE 1,1: PRINT "                                    "
GOTO OPM2: REM Возврат для ожидания нажатия клавиши
```

Пример 4.2.4 : **Обработка пунктов меню оператором SELECT CASE**

Оператор SELECT CASE, так же как и многострочный IF позволяет сформировать хорошо читаемую программу.

```
OPM3: REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
SELECT CASE nkl : REM nkl - переключающее выражение
   CASE 1  : REM 1 - условие выборки
     GOSUB M1  : REM M1, M11 - подпрограммы, вызываемые
     GOSUB M11 : REM при выполнении первого пункта меню
```

```
    GOSUB MENU
  CASE 2
    GOSUB M2
    GOSUB M21
    ...
  CASE ELSE
      LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
      BEEP 1: REM Выдача звукового сигнала
      SLEEP 3: REM Остановка на 3 секунды и очистка строки сообщения
      LOCATE 1,1: PRINT "                                    "
      GOTO OPM1: REM Возврат для ожидания нажатия клавиши
  END SELECT
```

Пример 4.2.5: **Подпрограмма выбора пунктов меню**

```
ONK: REM Подпрограмма обработки нажатия клавиши
msg1=" Введите правильно номер требуемого пункта меню "
LOCATE 23,1: REM Перемещение курсора в 23 строку
c$=SPACE$(80): PRINT c$: REM Очистка 23 строки
a=(40-LEN(msg1))/2:   REM центрирование сообщения
LOCATE 23,10:?TAB(a);msg1 :REM Вывод сообщения в 23 строку
1000 IF INKEY$="" THEN GOTO 1000: REM Пустой цикл до нажатия
        REM любой клавиши
nkl=VAL(c$)
RETURN
```

4.3. Остановки в программе

В ряде случаев при выполнении программы необходимо задержать ее выполнение на некоторое время, например, чтобы просмотреть результаты промежуточных вычислений, вывести сообщение программы, необходимое пользователю, и так далее. Такие остановки можно реализовать с помощью подпрограмм.

Пример 4.3.1:

```
OSTANOWKA1: REM остановка до нажатия любой клавиши
LOCATE 23,1: PRINT "Для продолжения нажмите любую клавишу"
1000 IF INKEY$="" THEN GOTO 1000 : REM пустой цикл
RETURN
```

Пример 4.3.2 :

```
OSTANOWKA2: REM остановка до нажатия любой клавиши
LOCATE 23,1: PRINT "Для продолжения нажмите любую клавишу"
A$=INPUT$(1) : REM ожидание ввода одного символа
RETURN
```

Пример 4.3.3:

```
OSTANOWKA3: REM остановка на заданное время
LOCATE 23,1 :  REM комментарий отсутствует
SLEEP 5 :      REM остановка на 5 секунд
RETURN
```

Пример 4.3.4:

```
OSTANOWKA4: REM остановка на заданное время
LOCATE 23,1: PRINT " Ждите. Идут вычисления"
REM функция TIMER хранит количество секунд, прошедших после полуночи
T=TIMER: REM переменной t присвоено текущее время
1000 IF TIMER-T<5 THEN GOTO 1000: REM остановка на 5 секунд
RETURN
```

5. АЛГОРИТМЫ РЕШЕНИЯ ТИПОВЫХ ЗАДАЧ

5.1. Исследование функции на отрезке

Исследование функции на отрезке включает следующие этапы (процедуры):

1. Анализ функции. Определение области определения функции.
2. Поиск точек разрыва функции, точек пересечения ее с осями координат и вертикальных асимптот, если они существуют.
3. Установление четности (нечетности), периодичности функции.
4. Определение критических точек. К критическим точкам относятся граничные точки и точки экстремумов.
5. Исследование функции на монотонность.
6. Определение интервалов выпуклости и вогнутости, точек перегиба.
7. Поиск асимптот графика функции.
8. Построение графика функции.

Первый этап неформализованный. По виду функции устанавливают область ее определения и точки разрыва, а также периодичность функции. Остальные этапы могут быть выполнены с помощью программы.

Вопросы исследования функций подробно рассмотрены в учебной литературе по высшей математике, например [1]. Поэтому рассмотрим лишь один вопрос - численное определение производной.

Производная от функции может быть определена численными методами. Известно, что

$$f(x) = \lim_{x \to \pm 0} \frac{(f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + 0(x), \tag{4.1}$$

отсюда вытекает способ численного дифференцирования. Если заменить предел Δx его конечным значением h, то получим приближенные формулы для вычисления первой и второй производных:

$$f'(x) \approx \frac{f(x+h) - (f(x-h)}{2h} \tag{4.2}$$

$$f''(x) \approx \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \tag{4.3}$$

Эти выражения представляют собой усеченные интерполяционные многочлены (многочлен Стирлинга). Одной из серьезных проблем в данном случае является выбор величины шага h. При уменьшении шага уменьшается ошибка усечения, но возрастает ошибка округления при вычислении производной. Поэтому стремятся выбрать оптимальную величину шага, при которой ошибка усечения и ошибка округления будут примерно равны. Для формулы (4.2) оптимальный шаг определяется из выражения [6]

$$h = \sqrt[3]{\frac{3\varepsilon}{M_3}} \quad \text{или} \quad \frac{1}{6h}\left|\Delta^3 y\right| \approx \frac{1}{2}\frac{\varepsilon}{h}, \tag{4.4}$$

где h - шаг, $\Delta^3 y$ - конечная разность 3-го порядка, ε - абсолютная погрешность вычисления функции, M_3 - максимальное значение конечной разности 3-го порядка. Таким образом, ошибка усечения равна примерно половине ошибки округления. Полная погрешность не превзойдет при этом 0.5 ε/h.

Для формулы (4.3) величина шага определяется из следующего соотношения

$$\frac{1}{12\,h^2}\left|\Delta^4 y\right| \approx 4\frac{\varepsilon}{h^2}. \tag{4.5}$$

Ввиду сложности самой процедуры отыскания оптимальной величины шага при вычислении производной функции в курсовом проекте рекомендуется принять значение шага в пределах от 0.1 до 0.01 без вычислений.

Чтобы определить точки пересечения графика функции с осями координат необходимо принять y=0 для определения точек пересечения с осью X и x=0 - для определения точек пересечения с осью Y.

При определении наличия четности (нечетности) следует исходить из того, что у четной функции F(x)=F(-x) для любого x.

Для исследования функции по п. 4-6 необходимо протабулировать на заданном отрезе функцию, ее первую и вторую производные. Результаты табулирования рекомендуется записать вначале в массив (массивы).

Вычисление пределов для определения асимптот можно проводить в одной из математических систем, например, DERIVE, EUREKA или MERCURY.

5.2. Решение алгебраических и трансцендентных уравнений

Алгоритм решения алгебраических и трансцендентных уравнений включает два самостоятельных этапа:

1. Отделение корней на заданном интервале или области определения функции;
2. Уточнение значения корней на отрезках отделения.

Отделение корня

Общая схема отделения корней уравнения на заданном интервале включает следующую последовательность операций:

- определить критические точки: точки экстремумов и граничные точки отрезка;
- вычислить значения функции в критических точках и составить таблицу смены знака функции;
- определить отрезки отделения корня.

При решении задачи с помощью ЭВМ алгоритм может быть несколько иным: выполнить табулирование функции на заданном интервале. В процессе табулирования необходимо:

- вычислять значения функции в текущей $F(x_i)$ и следующей $F(x_{i+1})$ тохах. Вычислить произведение этих функций;
- зафиксировать точки x_i и x_{i+1}, в интервале которых произведение функций будет отрицательным. Эти точки и будут границами отрезков отделения.

Уточнение значения корня на отрезке отделения

Уточнение значения корня на отрезке отделения осуществляется одним из методов, например, методом итераций, методом дихотомии или методом Ньютона (метод касательных). Алгоритмы реализации этих методов рассмотрены в [3 и 4].

Самым простым является алгоритм уточнения значения корня методом деления отрезка пополам (метод дихотомии):

1. Вычислить значение функции в точке ***a***: Ya=F(a).
2. Найти середину отрезка [a, b] - точку ***c***. Вычислить значение функции в точке ***c***: Yc=F(c).
3. Проверить, если значение функции в точке ***c*** меньше требуемой точности ***ε***, то ***c*** – корень. Вывести результат на печать. Иначе уточнить, где находится корень – слева или справа от точки ***c***.
4. Если произведение значений функций Ya и Yc, меньше 0,(корень находится слева от точки ***c***), то ***b=c*** и повторить операции по п. п. 2, 3, 4. Иначе Ya=Yc, ***a=c*** и повторить операции по п.п. 2, 3, 4.

5.3. Вычисление определенного интеграла

При выполнении курсовой работы по данной теме необходимо вычислить определенный интеграл с заданной точностью двумя или тремя способами и сравнить полученные результаты по точности достигаемых результатов при заданном числе шагов или по числу шагов необходимых для достижения заданной точности. Для вычисления интеграла можно использовать методы средних, правых или левых прямоугольников, метод трапеций или метод Симпсона, которые подробно описаны в литературе.

Вычисление интегралов осуществляется по следующим формулам:

для метода средних прямоугольников

$$\int_a^b f(x)dx = \sum_{i=1}^{n} h_i f(x_{i-1/2}); \qquad (4.6)$$

для метода трапеций

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i (y_{i-1} + y_i); \qquad (4.7)$$

для метода Симпсона

$$\int_a^b f(x)dx \approx \frac{h}{3}[y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n]. \qquad (4.8)$$

Здесь x_i - текущее значение аргумента; $(x_{i-1/2})$ - значение аргумента в середине отрезка разбиения (полуцелая точка); y_i - значение функции при соответствующем значении аргумента; h - шаг разбиения отрезка интегрирования, h=n/2; n - число отрезков разбиения интервала интегрирования. Начальное значение n должно быть четным, например 4.

При вычислении определенного интеграла с заданной точностью необходимо организовать два вложенных цикла. Внутренний цикл служит для вычисления площади криволинейной трапеции при текущем значении шага разбиения h отрезка интегрирования, а также для вычисления суммарной ошибки численного интегрирования, внешний цикл обеспечивает достижение заданной точности интеграла путем последовательного уменьшения шага разбиения. Вычислительный процесс прекращается, когда величина ошибки интегрирования будет меньше заданного значения.

Обычно число отрезков разбиения N увеличивается на каждом шаге в два раза, что обеспечивает быстрое достижение заданной точности.

Погрешность численного интегрирования R равна сумме погрешностей численного интегрирования R_i на каждом отрезке $[x_{i-1}, x_i]$, которые определяется по следующим формулам:

$$R_I = \frac{1}{24} h_i^3 f^{(4)}(x_{i-1/2})$$ - для формулы средних прямоугольников; (4.9)

$$R_i = \frac{1}{12} h_i^3 f^{(2)}(x_i)$$ - для формулы трапеций; (4.10)

$$R_I = \frac{h^4}{180} f^{(4)}(x)$$ - для формулы Симпсона. (4.11)

Вычисление погрешности интегрирования можно оформить в виде функций, определяемых пользователем.

В практических целях такой алгоритм вычисления интеграла неэффективен. Гораздо проще задать требуемую точность и проводить вычисление интеграла одним из упомянутых выше способов. Условием окончания процесса вычисления будет выполнение неравенства $(I - I_1)/3 < \varepsilon$, где I и I_1 - значение интеграла на текущем и предыдущем шаге, соответственно.

5.4. Построение изображения детали

Курсовая работа на данную тему не содержит математических моделей и вычислительных алгоритмов и с этой точки зрения является достаточно простой. При выполнении данной работы студент должен продемонстрировать умение использовать графические возможности языка программирования, стандартные программы построения меню, обработки пунктов меню и остановок в программе.

Программа построения изображения детали должна содержать следующие подпрограммы:

- изображение детали в аксонометрии;
- изображение детали в аксонометрии с разрезом;
- проекции детали;
- перемещение детали в указанную точку экрана;
- поворот детали на заданный угол;
- изменение масштаба изображения.

Для обеспечения универсальности программы рекомендуется ввод данных о размерах детали организовать с помощью операторов DATA, READ, RESTORE. Данные считывать в массивы переменных. При использовании оператора DRAW для построения, например, проекций также рекомендуется при описании фигуры вместо констант использовать переменные.

Все подпрограммы по изменению масштаба, повороту, перемещению детали должны иметь режим диалога для запроса требуемых параметров и контроль выхода изображения за границы экрана. Каждая экранная форма должна содержать текстовое сообщение о порядке действий пользователя и, при необходимости, подменю для управления изображением на экране.

Одним из сложных алгоритмов является алгоритм поворота детали на заданный угол с переносом в указанную точку. Ниже приводится алгоритм поворота детали вокруг оси Z. Поворот детали вокруг осей X или Y требует более сложных алгоритмов.

Пример 5.4.1: **Алгоритм поворота изображения на заданный угол с переносом в заданную точку**

```
500 REM Алгоритм поворота изображения с переносом
505 REM в заданную точку
510 GOSUB 200:REM Вызов подпрограммы построения изображения
520 x1=190: y1=80: x2=400: y2=310:REM Ввод координат
                    REM изображения на экране
```

```
530 REM Определение размерности массива
  n=0: REM текущая переменная, размерность массива
  LOCATE 23,25:PRINT "ждите, идет считывание изображения"
  FOR i=x1 TO x2
  FOR j=y1 TO y2
    IF POINT (i,j)=1 THEN n=n+1
  NEXT j
  NEXT i
  DIM ax(n), ay(n)
540 REM Считывание данных в масссив
  l=0
  FOR i=x1 TO x2
  FOR j=y1 TO y2
    IF POINT (i,j)=1 THEN l=l+1: ax(l)=i-x1:ay(l)=j-y1
  NEXT j
  NEXT i
550 REM Воспроизведение изображения
  CLS
  INPUT "Введите координаты точки переноса X и Y";x0,y0
  INPUT "Введите угол поворота в градусах";f1
  f=f1*pi/180: REM перевод градусов в радианы
  CLS
  PSET (x0,y0)
  FOR i=1 TO n
    bx=ax(i): by=ay(i)
    a=x0 + bx*cos(f)- by*sin(f)
    b=y0 + bx*sin(f)+ by*cos(f)
    PSET (a,b)
  NEXT i
```

Изменение масштаба изображения реализуется легко путем переопределения математических координат с помощью оператора WINDOW (0,0)-(640*mx,200*my), где mx и my - масштабы по соответствующим осям.

Чаще всего расстояние между пикселями на экране монитора по горизонтали и по вертикали разные, поэтому при изображении дуг и окружностей возникают искажения. Чтобы избавиться от этих искажений, необходимо установить строгое соответствие между значениями размеров экрана по горизонтали и по вертикали: X=4/3Y. С учетом сказанного оператор WINDOW следует применять в следующем виде: WINDOW(0,0)-(X,Y).

В программе можно предусмотреть демонстрацию возможностей программы.

5.5. Моделирование движения механизма

Под моделированием движения механизма понимается воспроизведение положения механизма в последовательные моменты времени.

При выполнении курсового проекта на данную тему необходимо решить следующие вопросы:

1. Разработать математическую модель механизма (см. [5]);
2. Разработать алгоритмы обеспечивающие:

- воспроизведение положения механизма в исходном состоянии;
- воспроизведение положения механизма в нескольких промежуточных точках;
- определение зон движения механизма;
- определение области, занимаемой механизмом в процессе движения;
- построение траектории движения заданной точки.

При наличии у механизма только ползунов или катков, у которых звенья механизма присоединены к центру катка, никаких принципиальных трудностей в разработке математической модели не возникает. Для вывода математических зависимостей используются теоремы синусов, косинусов, свойства прямоугольных треугольников, подобия фигур и тому подобные соотношения, известные из школьного курса математики.

Сложнее обстоит дело с механизмами, имеющими катки, у которых звенья механизма присоединены к точкам на поверхности катка. Рассмотрим в качестве примера механизм, представленный на рис. 4.1.

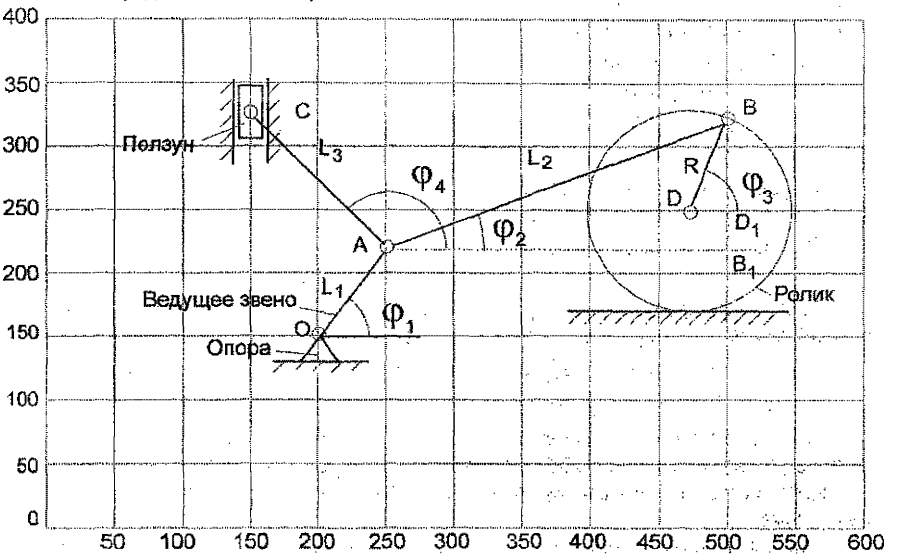


Рис. 4.1. Механизм

Здесь l_2 - длина шатуна AB; r - радиус катка. Первые четыре уравнения получают из рассмотрения прямоугольников ABB_1 и DBD_1. Пятое уравнение составляется из следующих соображений: координаты точки B лежат но поверхности окружности, поэтому можно записать уравнение окружности с центром в точке D и радиусом, равным радиусу катка r.

1. Выберем масштаб изображения из условий, что размер по оси X –ов больше размера по оси Y-ов в 4/3 (иначе возникают нелинейные искажения, связанные с параметрами экрана) и изобразим механизм в масштабе.
2. Определим параметры узлов и составим таблицу исходных данных:

№ узла	X	Y	L	φ
O	200	150	L1=90	55
A	Xa	Ya	L2=240	
B	Xb	Yb	R=75	
C	150	Yc	L3=150	
D	Xd	170		

3. Составим математическую модель

$$x_a = x_0 + l_1 \mathrm{Cos}(\varphi_1) \quad (4.12)$$

$$y_a = y_0 + l_1 \mathrm{Sin}(\varphi_1) \quad (4.13)$$

На данном рисунке каток имеет две степени свободы. Он может вращаться вокруг своей оси и перемещаться в горизонтальном направлении. В исходных данных высота расположения катка и его радиус заданы, а следовательно, и вертикальная координата центра катка Y_d известна. Горизонтальное положение катка X_d - неизвестно. Наибольшие трудности возникают при определении значения угла φ_2.

Для определения значения угла φ_2 необходимо составить и решить систему из пяти уравнений:

$$\begin{cases} x_b = x_a + l_2\cos\varphi_2 & (4.14) \\ x_b = x_d + r\cos\varphi_3 & (4.15) \\ y_b = y_a + l_2\sin\varphi_2 & (4.16) \\ y_b = y_d + r\sin\varphi_3 & (4.17) \\ (x_b - x_d)^2 + (y_b - y_d)^2 = r^2 & (4.18) \end{cases}$$

Из уравнений 4.14 и 4.15 находим значение X_d:

$$x_d = x_a + l_2\cos\varphi_2 - r\cos\varphi_3 \qquad (4.19)$$

Из уравнений 4.16 и 4.17 находим значение угла φ_3:

$$\varphi_3 = \arcsin((y_a + l_2\sin\varphi_2 - y_d)/r) \qquad (4.20)$$

Заменив в выражении (4.17) угол φ_3 на его значение из (4.20) и подставив значения x_b (4.16), x_d и y_b (4.17) в (4.20), получим уравнение

$$((x_a + l_2\cos\varphi_2 - (x_a + l_2\cos\varphi_2 - r\cos(\arcsin((y_a + l_2\sin\varphi_2 - y_d)/r)))^2 + \\ + (y_a + l_2\sin\varphi_2 - y_d)^2) - r^2 = 0. \qquad (4.21)$$

В выражении 4.21 один неизвестный параметр - угол φ_2. Найти значение угла φ_2 из данного уравнения можно с помощью одного из методов параметрической оптимизации, например, методом итераций. Фрагмент программы, реализующий данный метод, приведен ниже.

Для вычисления координат точки С следует воспользоваться теоремой Пифагора, так как после определения координат точки А будут известны катет и гипотенуза треугольника ACC_1.

Пример 5.5.1: **Программа для подбора значения угла φ_2.**

```
REM Программа построения механизма
SCREEN 9
Y=400: X=4/3*Y
WINDOW (0,0)-(X,Y): REM Переопределение координат на математические
REM Ввод исходных данных
pi=ATN(1)*4
e=1   'Точность поиска
x0=100: y0=100
yd=150
l1=40: l2=175:l3=50
i1=2*pi/3
REM Цикл изменения значения угла φ1
for i=0 to i1 step 0.3
f1=i
xa=x0+l1*cos(f1): ya=y0+l1*sin(f1)
s1=e+1:df=.01: f2=pi/10
  REM Цикл изменения значения угла φ2
  while abs(s1)>e and f2<pi/2
```

```
    a=(ya-yd+l2*sin(f2))/l3
    b=a*l3
    if (1-a^2)<=0 then goto m1
    s1=l3^2-((xa+l2*cos(f2)-(xa+l2*cos(f2)-l3*cos(atn(a/sqr(1-a^2)))))^2+b^2)
    goto m2
    m1:
     ?"нет решения, F2="f2*180/pi,"F1="f1*180/pi
    m2:
    ft=f2
  f2=f2+df
  wend
    REM Расчет координат
    f2=ft
    a=(ya-yd+l2*sin(f2))/l3
    if (1-a^2)<0 then goto m3
    f3=atn(a/sqr(1-a^2))
    xb=xa+l2*cos(f2)
    yb=ya+l2*sin(f2)
    xd=xb-l3*cos(f3)
  m3: REM Построение рисунка
  circle (x0,y0),4
  line (x0,y0)-(xa,ya)
  circle (xa,ya),4
  line (xa,ya)-(xb,yb)
  circle (xb,yb),2
  circle (xd,yd),l3
  next i
end
```

Указания к решению задач 2.2, 2.7, 2.14,2.28, 2.29

В задачах 2.2, 2.7, 2.14, 2.28, 2.29 необходимо в исходных данных указать положение точки М (рис. 4.2). Написать уравнение прямой, проходящей через точку М:

$$y=kx+b \qquad (4.22)$$

Для точки М в этом уравнении известны значения X и Y , $k=tg(\alpha)$. Поэтому можно вычислить значение свободного члена b.

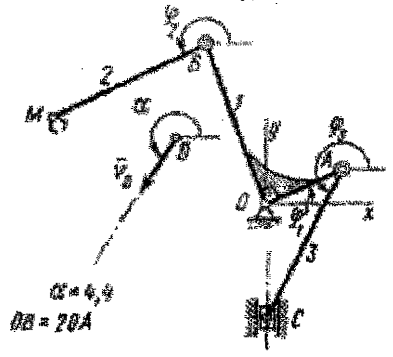


Рис. 4.2. Механизм

Точка М в процессе движения должна постоянно находиться на заданной прямой. Ее уравнение также будет иметь вид $Y_m=kX_m+b$, в котором будет две неизвестных X_m и Y_m. Кроме того, в любой момент времени ее расстояние от точки В должно быть равно длине звена ВМ, длина которого может быть определена через координаты точек В и М по теореме Пифагора. Таким образом получаем систему уравнений с двумя неизвестными:

$$\begin{cases} Y_m=kX_m+b \\ BM^2=(Xm-Xb)^2+(ym-Yb)^2 \end{cases},$$

которое легко решается.

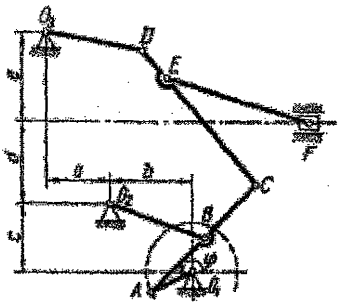


Рис. 4.3. Механизм

Указания к решению задач Приложения 3

Рассмотрим порядок нахождения координат точки В на рис. 4.3. У точки В не известны ни x ни y. Но известны координаты точек А и O_2.

Определим координаты точки В через координаты точки А и С (рис. 4.4).

$$x_b=x_c+BC\cos(\varphi_3) \quad (1)$$

$$y_b=y_c-BC\sin(\varphi_3) \quad (2)$$

$$_b=x_a+AB\cos(\varphi_2) \quad (3)$$

$$y_b=y_a+AB\sin(\varphi_2) \quad (4)$$

Решим эту систему уравнений. Освободимся от переменных x_b и y_b. Из уравнений (1) и (3) получим:

$$BC\cos(\varphi_3)= x_a - x_c + AB\cos(\varphi_2). \quad (5)$$

Из уравнений (3) и (4) получим:

$$BC\sin(\varphi_3)= y_a - y_c + AB\sin(\varphi_2). \quad (6)$$

Возведем выражения (5) и (6) в квадрат и сложим их, тогда получим следующее выражение:

$$BC^2\cos(\varphi_3)^2+ BC^2\sin(\varphi_3)^2= (x_a - x_c + AB\cos(\varphi_2))^2+(y_a - y_c + AB\sin(\varphi_2))^2. \quad (7)$$

Воспользуемся известным выражением $Cos(x)^2+Sin(x)^2=1$, раскроем скобки и приведем подобные члены:

$$AB^2-BC^2+(xa-xc)^2+(ya-yc)^2+2AB(xa-xc)Cos(\varphi_2)+2AB(ya-yc)Sin(\varphi_2)=0 \quad (8)$$

Введем вспомогательную переменную $U=(AB^2-BC^2+(xa-xc)^2+(ya-yc)^2)/(2AB)$, повторно разделим неизвестные переменные $Cos(\varphi_2)$ и $Sin(\varphi_2)$ и повторим операцию по п. 6 и 8. В результате получим:

$$U^2+2U(xa-xc)Cos(\varphi_2)+ (xa-xc)^2Cos^2(\varphi_2)= (ya-yc)^2Sin^2(\varphi_2)$$

$$U^2 -(ya-yc)^2 +2U(xa-xc)Cos(\varphi_2)+ ((xa-xc)^2 + (ya-yc)^2)Cos^2(\varphi_2)= 0 \quad (9)$$

Выражение (9) является квадратным уравнением относительно $Cos(\varphi_2)$ и может быть решено аналити-чески. После вычисления значения $Cos(\varphi_2)$ находим значение $Sin(\varphi_2)$ и координаты точки В согласно выражений (3), (4).

Аналогично вычисляются значения координат точки D на рис. 4.3.

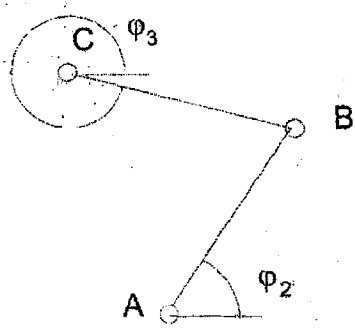
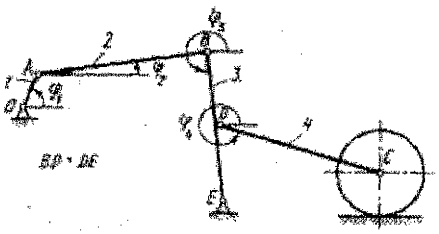


Рис.4.4. Фрагмент механизма рис. 4.3.

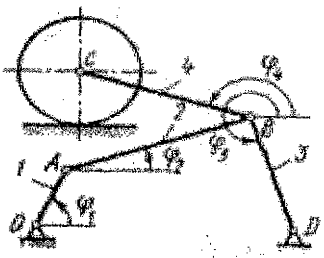
6. Приложения

Приложение 1

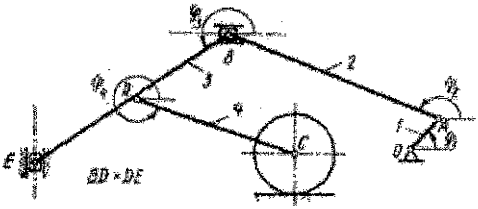
1



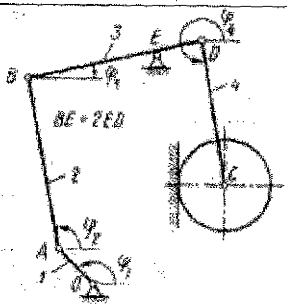
2



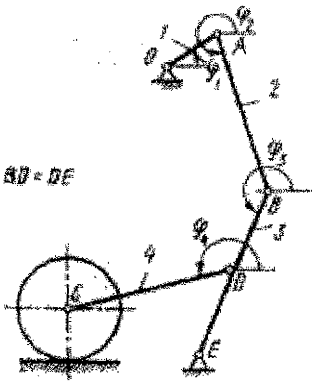
3



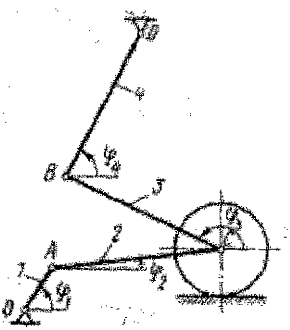
4



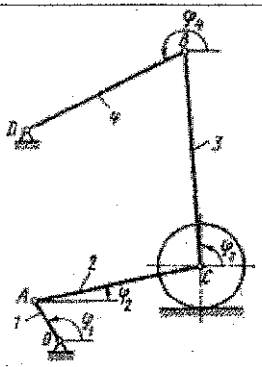
5



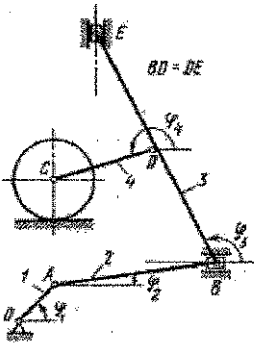
6



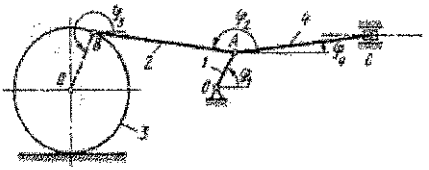
7



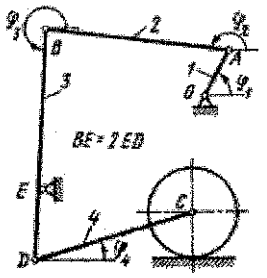
8



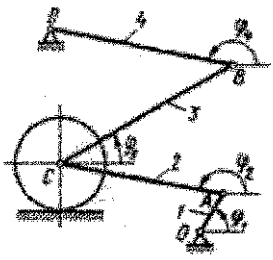
9



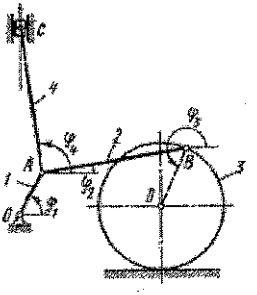
10



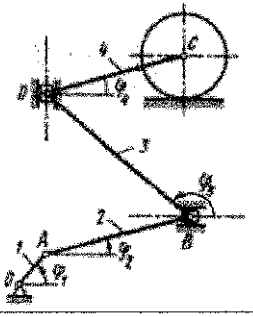
11



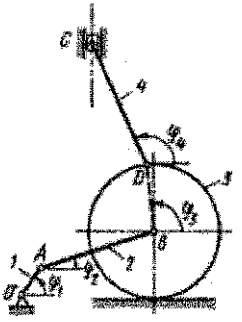
12



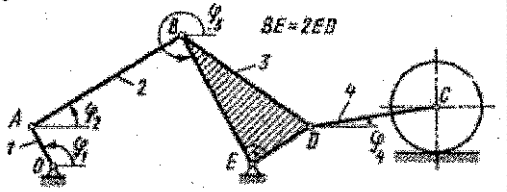
13



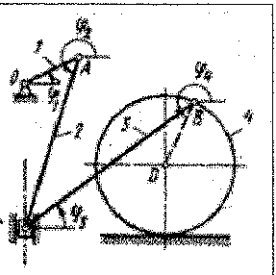
14



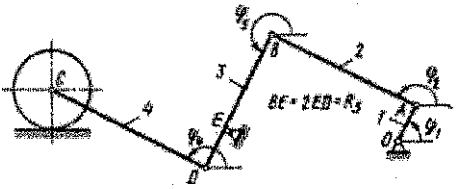
15



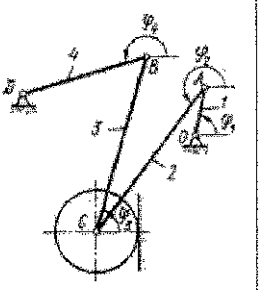
16



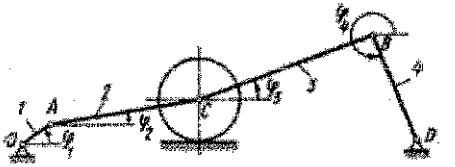
17



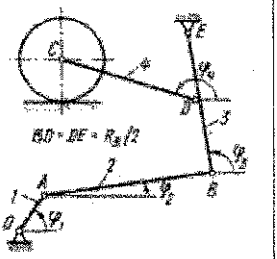
18



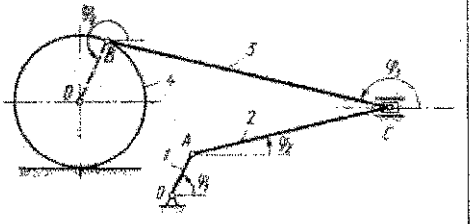
19



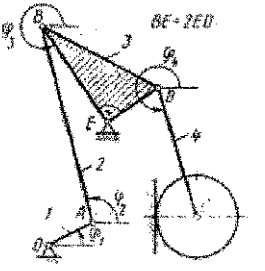
20



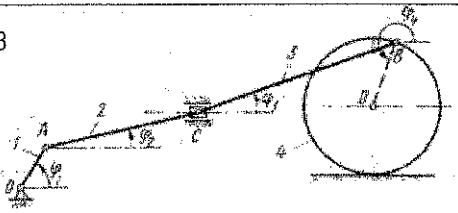
21



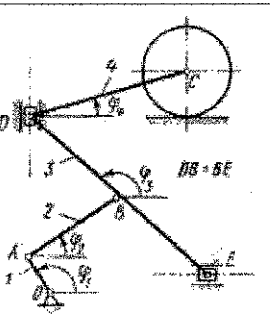
22



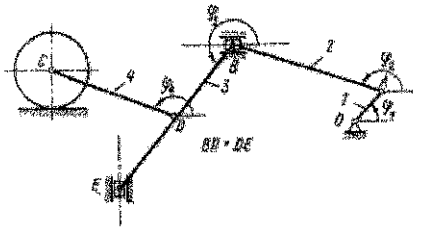
23



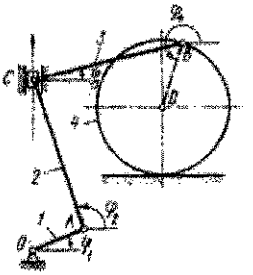
24



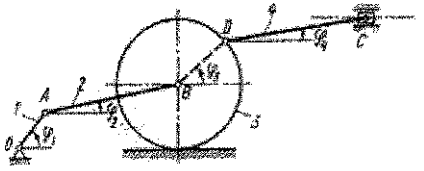
25



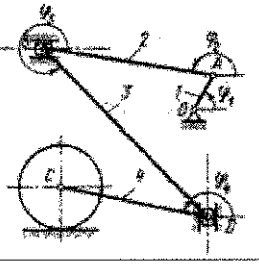
26



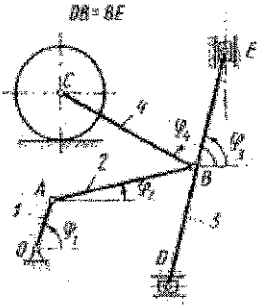
27



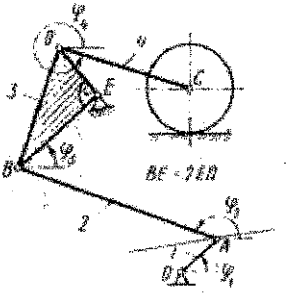
28



29

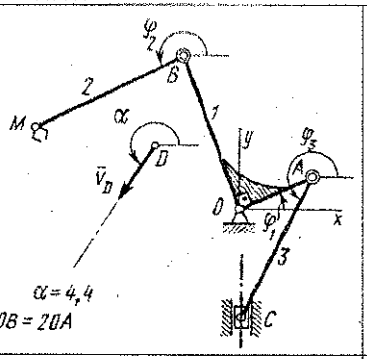


30

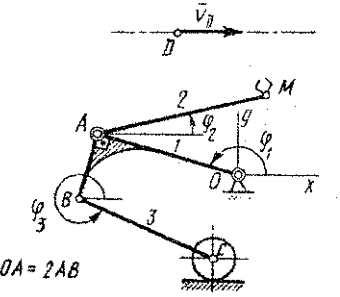


Приложение 2

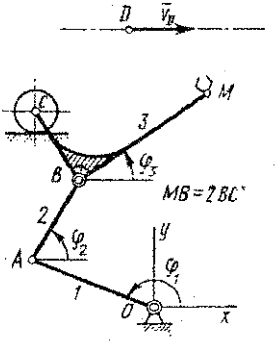
1



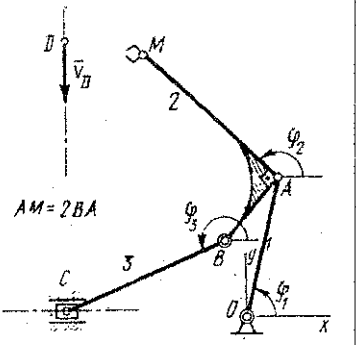
2



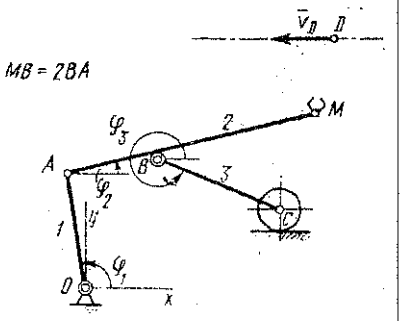
3



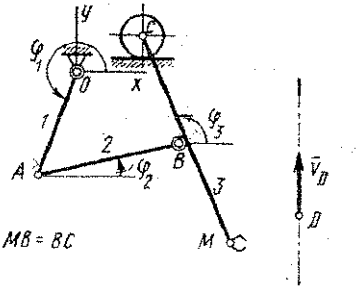
4



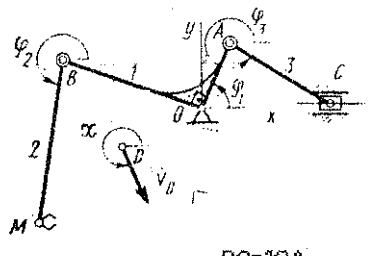
5



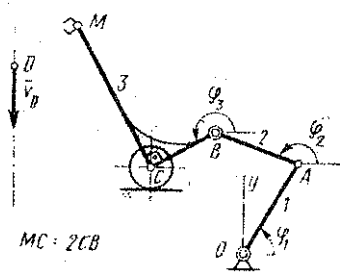
6



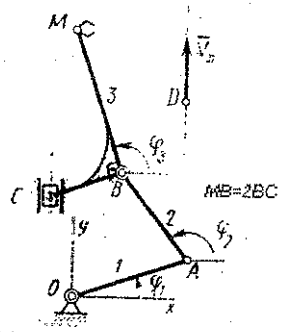
7



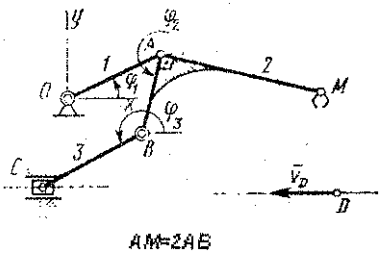
8



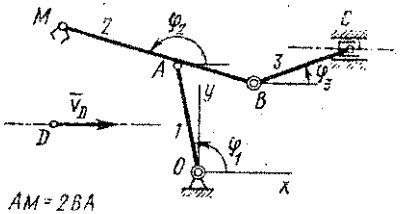
9



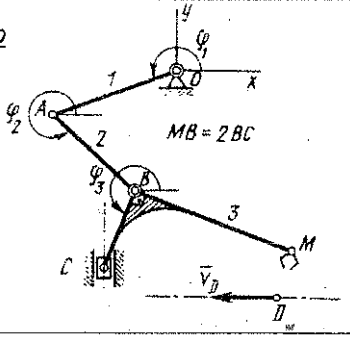
10



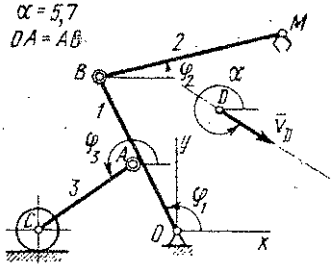
11



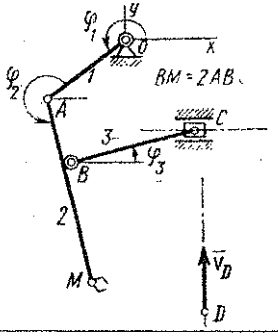
12



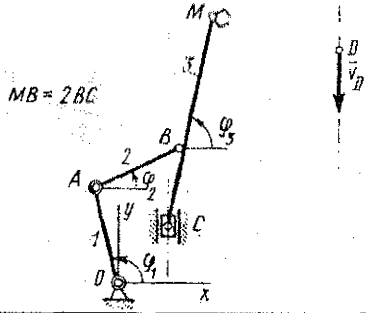
13



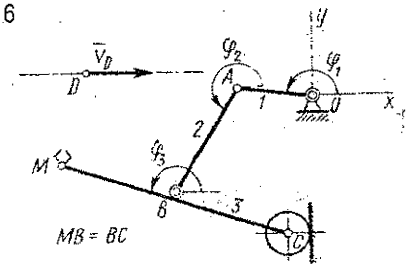
14



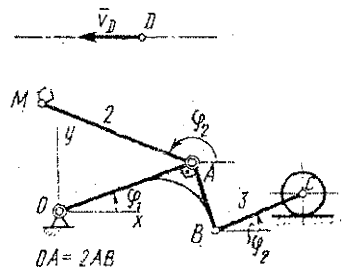
15



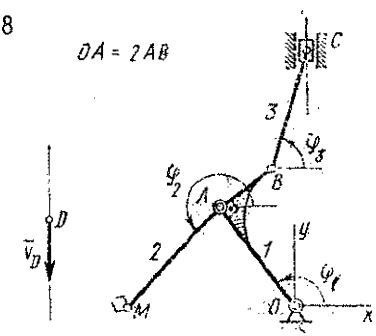
16



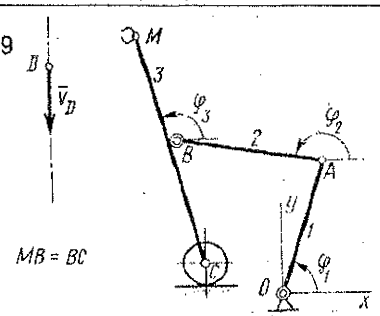
17



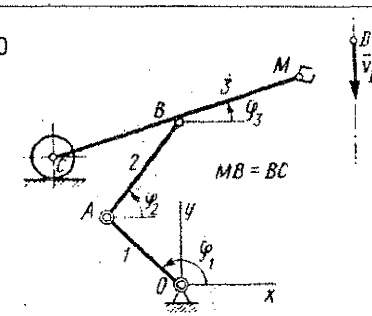
18



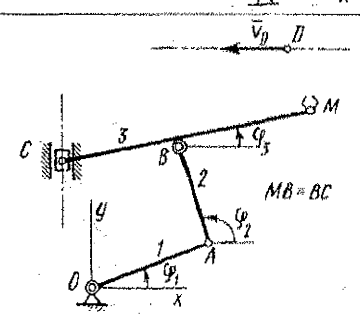
19



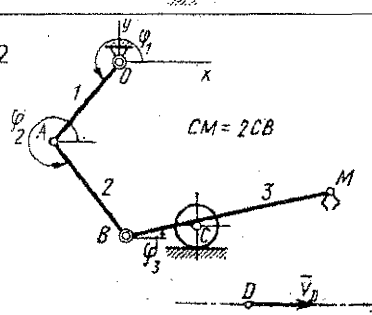
20



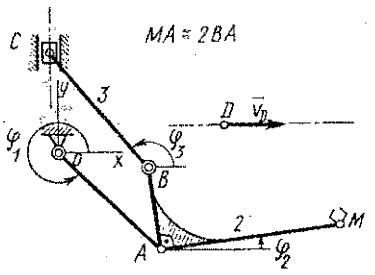
21



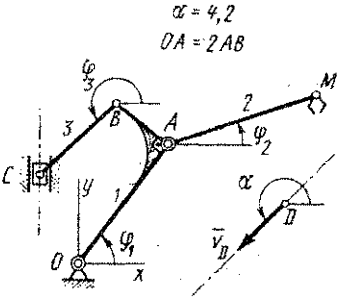
22



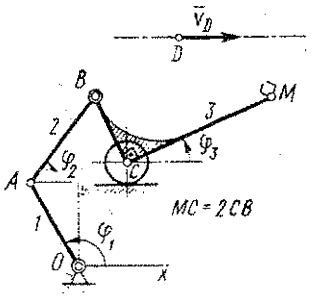
23



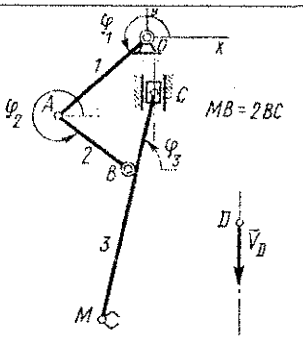
24



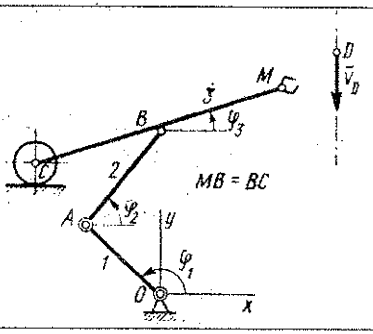
25



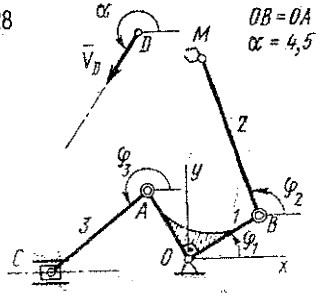
26



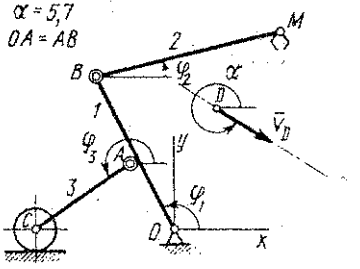
27



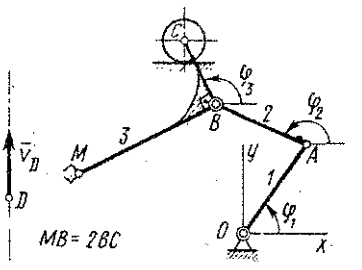
28



29

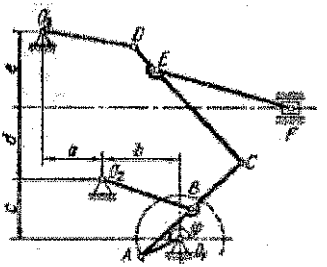


30

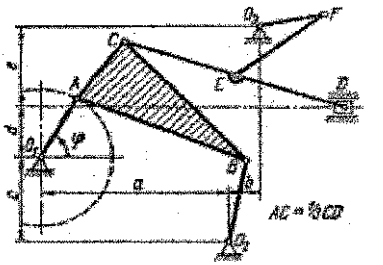


Приложение 3

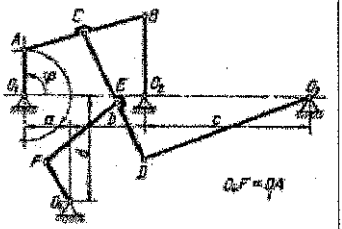
1



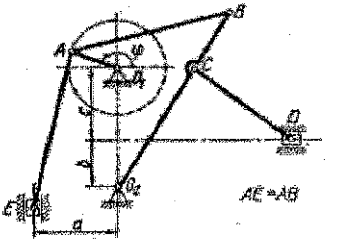
2



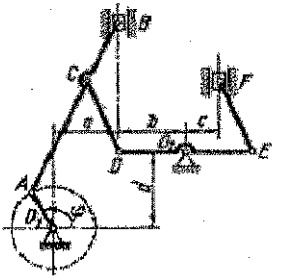
3



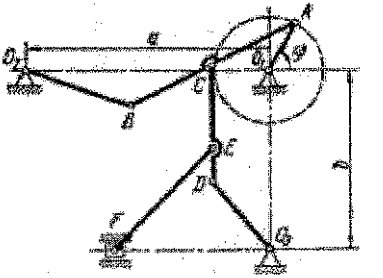
4



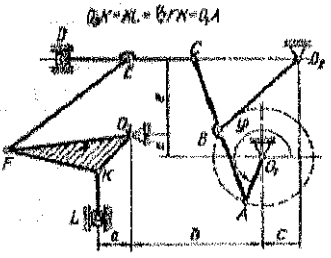
5



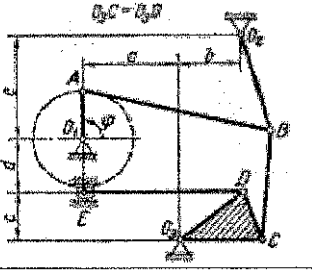
6



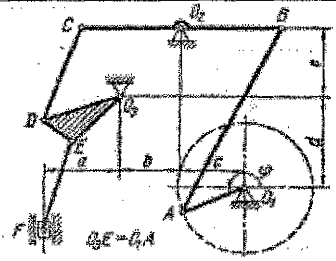
7



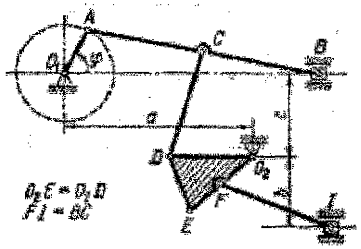
8



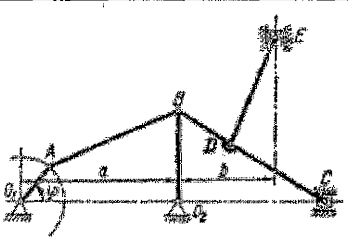
9



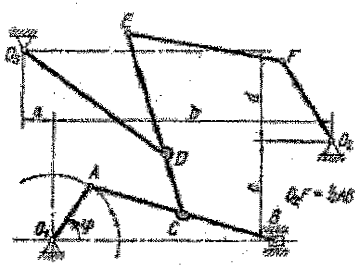
10



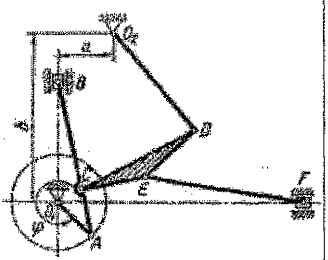
11



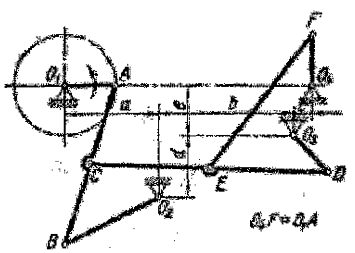
12



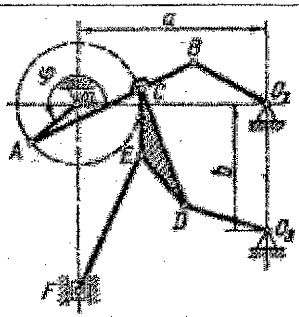
13



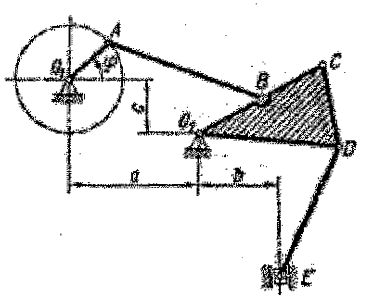
14



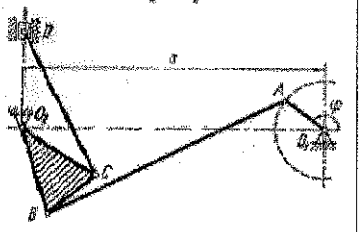
15



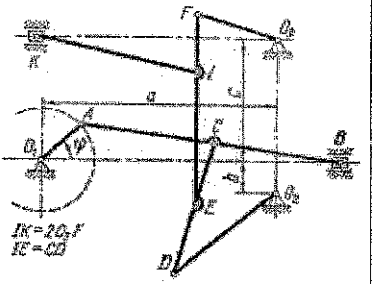
16



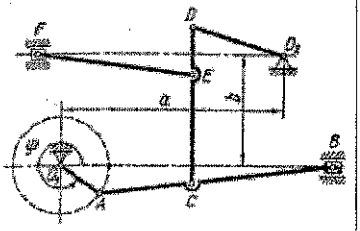
17



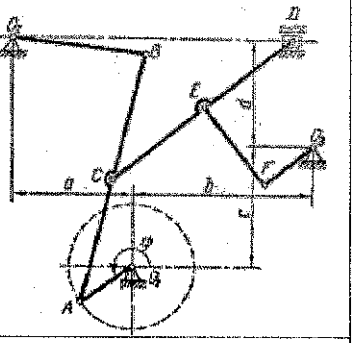
18



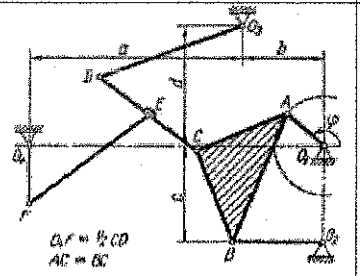
19



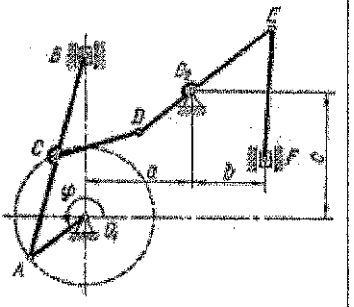
20



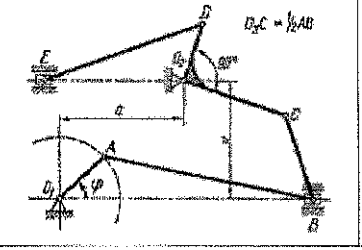
21



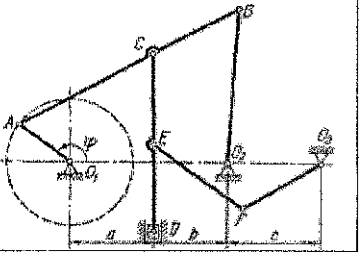
22



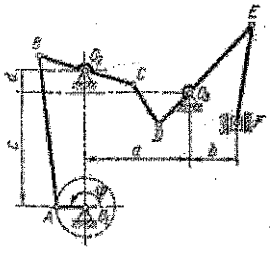
23



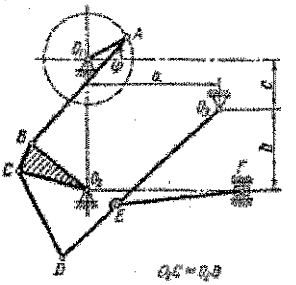
24



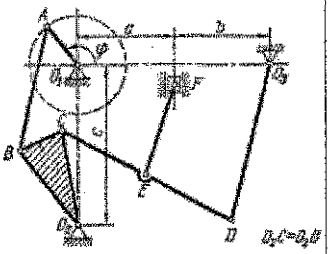
25



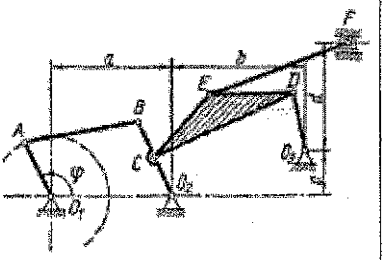
26



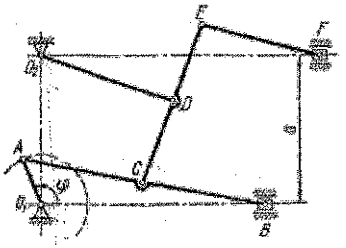
27



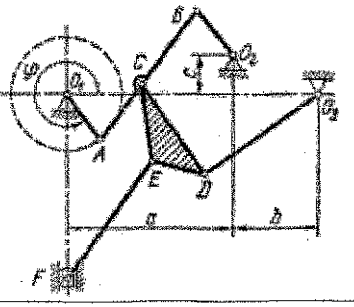
28



29



30



СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Сборник индивидуальных заданий по высшей математике, ч.1.- Мн.: Высшая школа, 1990. - 270 с.

2. Воробьева Г. Н. , Данилова А. Н. Практикум по вычислительной математике. - М.: Высшая школа, 1990. - 208 с.: ил.

3. Турчак Л. И. Основы численных методов. - М.: Наука, 1987. - 320 с.

4. Дьяконов В. П. Справочник по алгоритмам и программам на языке бейсик для персональных ЭВМ. - М.: Наука, 1989. - 240 с.

5. Вычислительная техника, программирование и математическое моделирование: методические указания - Брест, БрПИ, 1991. - 34 с.

6. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах. - М.: Наука, 1972, 368 с.: ил.

7. Кетков Ю. Л. GW-, Turbo- и Quick BASIC для IBM PC.- М.: Финансы и статистика, 1992.- 240 с.; илл.

Учебное издание

Составитель: Быков Вячеслав Леонидович

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению курсовых работ по дисциплине «Информатика»
для специальностей 36 01 01 «Технология машиностроения», 36 01 03
«Технологическое оборудование машиностроительного производства»,
27 01 06 «Техническая эксплуатация автомобилей»

Издание третье, дополненное

Ответственный за выпуск Быков В. Л.
Редактор Строкач Т. В.
Корректор: Никитчик Е.В.
Компьютерная вёрстка: Кармаш Е.Л.

Подписано к печати 27.01.2005 г. Формат 60x84 1/16. Бумага «Снегурочка». Усл.п.л. 2,79. Уч.изд.л. 3,0. Тираж 150 экз. Заказ № 202. Отпечатано на ризографе учреждения образования «Брестский государственный технический университет». 224017, г. Брест, ул. Московская, 267.