Министерство образования Республики Беларусь

Брестский политехнический институт

Кафедра вычислительной техники и прикладной математики

Методические рекомендации

для выполнения курсовых работ по дисциплинам
"Вычислительная техника, программирование и математическое моделирование", "Вычислительная техника и программирование"

Брест 1999

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 Turbo-Basic.

Предназначены для студентов первого и второго курсов специальностей "Технология машиностроения", "Мелиорация и водное хозяйство", "Водоснабжение, водоотведение, очистка природных и сточных вод" по дисциплинам "Вычислительная техника, программирование и математическое моделирование", "Вычислительная техника и программирование" и имеют целью оказать помощь студентам в подготовке и оформлении курсовых работ по названным дисциплинам.

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. Требования к содержанию курсовой работы

Курсовая работа выполняется студентом самостоятельно с использованием языка программирования Бейсик (Паскаль и др.) или с использованием электронной таблицы SuperCalc 4.

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 10 - 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 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 задачи одномерной оптимизации одним или двумя способами (методы дихотомии, итераций, касательных) или интерполяции методами Ньютона и Лагранжа.

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.

Содержание курсовой работы: титульный лист, бланк задания, содержание, введение (не обязательно), пояснительная записка, заключение (не обязательно), список использованной литературы, приложение.

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.

Описание переменных представляется в виде таблицы рис.1.1.

Таблица 3.1

Описание переменных <- заголовок таблицы

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

Рис.1.1. Пример оформления таблицы

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

1.2. *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, перечень использованных операторов пишутся 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. ЗАПРЕЩАЕТСЯ распечатывать тексты программ через редактор текста WINWORD.

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 - 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

Брестский политехнический институт

Кафедра вычислительной техники и прикладной математики

Тема: *Исследование функции одной переменной*

Выполнил: студент группы № _____

_____

(Фамилия, инициалы)

Консультант:_____

(Фамилия, инициалы)

Проверил: преподаватель

(Фамилия, инициалы)

г. Брест, 1998 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 1,5a, допускается устанавливать b равное 2a.

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 проставляются 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.

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.

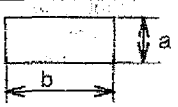
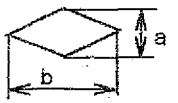
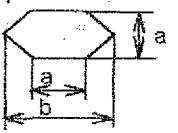
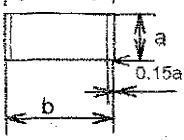
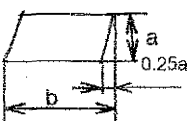
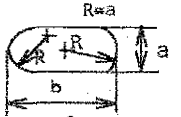
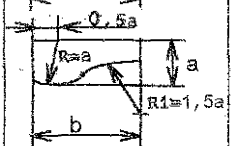
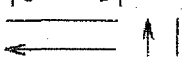
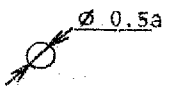
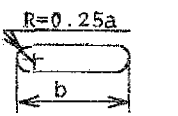
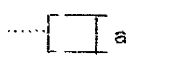
Математическая модель задачи представляет собой совокупность математических выражений, описывающих данную задачу.

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.

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, изменяющих программу
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 символами в пределах листа
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементами схемы с пояснениями

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 устранении ошибок, обнаруженных в процессе эксплуатации, доработке программы и ее совершенствовании в соответствии с требованиями заказчика.

2.2. Структура программы

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.

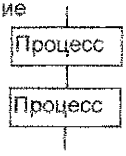
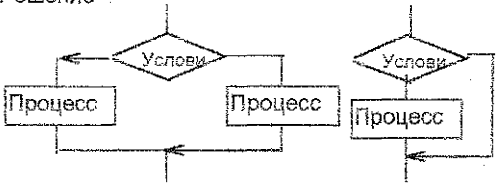
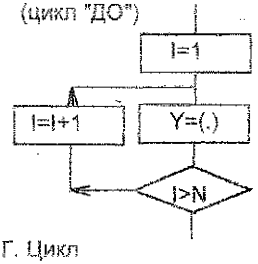
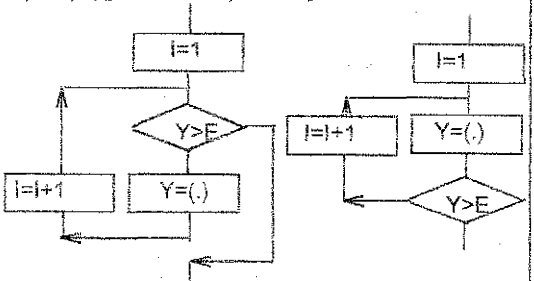
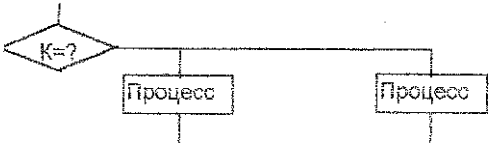
Этот процесс продолжается, пока схема алгоритма не будет доведена до базовых структур соответствующего языка программирования. Базовые элементы схем алгоритмов для всех языков программирования в основном совпадают. Отличия могут заключаться только в форматах используемых операторов. Основные базовые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

Таблица 2.1

Представление базовых структур алгоритмов

Схема алгоритма	Запись на алгоритмическом языке
А. Следование 	*Нач* <выражение> . <выражение> *Кон*
Б. Решение 	*Если* <условие> *то* <выражение> *иначе* <выражение>
В. Цикл с параметром или с повторением (цикл "ДО") 	*Для* I *от* N1 *до* N2 *шаг* N3 *нц* <выражение> . <выражение> *кц*
Г. Цикл а) с предусловием б) с *постусловием* 	*Пока* <условие> *нц* <выражение> . <выражение> *кц*
Д. Множественный переход 	*Выбор* при условии 1:<выраж.> . при условии N:<выраж.> *Все*

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.

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

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;

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

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

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

3. Типовые схемы алгоритмов

3.1. Ввод данных

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

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 обозначение [;/,] 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. 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:

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:

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.

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>

Примеры:

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 1: **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
          REM в одномерный массив
290 NEXT i
```

3.2. Паспорт 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 2: **Паспорт программы**

```
REM Паспорт программы
CLS:
SCREEN 2
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 обазования Республики Беларусь"
LINE(100,25)-(540,25)
```

```
LOCATE 5,20:?"   Брестский политехнический институт"
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 3: **Меню программы**

```
MENU:  REM Меню программы
DEFSTR c
REM  переменные, имя которых начинается с символа "c" -
REM  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
```

3.3. Обработка пунктов меню

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

Оператор **GOTO M** обеспечивает безусловный переход к указанной метке. В качестве метки может использоваться любой набор букв и цифр заканчивающийся двоеточием, например, **MENU1:**.

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

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

IF <условие> THEN <оператор перехода>

IF <условие> THEN <оператор вызова подпрограммы>

Пример 4: **Обработка пунктов меню оператором IF**

```
OPM1:  REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы обработки нажатия
            REM клавиш
     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
CLS
PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1:   REM Выдача звукового сигнала
DELAY 3:  REM Останов на 3 секунды
GOTO OPM1: REM Возврат для ожидания нажатия клавиши"
```

Здесь Mn - метка подпрограммы выполнения соответствующего пункта меню.

Пример 5: **Обработка пунктов меню оператором ON**

```
OPM2:  REM подпрограмма обработки пунктов меню
...
      Вариант 1                     |        Вариант 2
ON nkl GOTO  M1,M2,...,Mn           |  ON nkl GOSUB M1,M2,...,Mn
CLS
PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1:   REM Выдача звукового сигнала
DELAY 3:  REM Останов на 3 секунды
GOTO OPM2: REM Возврат для ожидания нажатия клавиши"
```

Пример 6: **Обработка пунктов меню оператором SELECT CASE**

```
OPM3:  REM подпрограмма обработки пунктов меню
...
SELECT CASE nkl :  REM nkl - переключающее выражение
  CASE nkl=1   :  REM nkl=1 - условие выборки
    GOSUB M1   :  REM M1, M11 ..- подпрограммы, вызываемые
    GOSUB M11  :  REM при выполнении первого пункта меню
    ...
  CASE nkl=2
    GOSUB M2
    GOSUB M21
    ...
  ELSE CASE
    CLS
    PRINT "Неправильный ввод. Введите пункт меню"
    BEEP 1:   REM Выдача звукового сигнала
    DELAY 3:  REM Останов на 3 секунды
    GOTO OPM3: REM Возврат для ожидания нажатия клавиши"
```

```
END CASE
```

Пример 7: **Подпрограмма обработки нажатия клавиши**

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

3.4. 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 8:

```
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 9:

```
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 10:

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

Пример 11:

```
OSTANOWKA4: REM остановка на заданное время
LOCATE 23,1: PRINT " Ждите. Идут вычисления"
t=TIME: REM переменной t присвоено текущее время
1000 IF TIME-t<5 THEN GOTO 1000: REM остановка на 5 секунд
RETURN
```

4. Алгоритмы решения типовых задач

4.1. 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 график функции.

Первый этап неформализованный. По виду функции определяют 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

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

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

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

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

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]

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

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 1.5 ε/h.

Для формулы (4.3) величина шага определяется из следующего соотношения

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

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.

4.2. 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.

4.2.1. 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 знак функции в предыдущей и текущей токах;
- фиксировать точки, в которых знаки функции имеют разные значения. Эти точки и будут границами отрезков отделения".

4.2.2. Уточнение значения корня на отрезке отделения

Уточнение значения корня на отрезке отделения осуществляется одним из методов, например, методом итераций, методом дихотомии или методом Ньюто-

на (метод касательных). Алгоритмы реализации этих методов рассмотрены в [3 и 4].

4.3. *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

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

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:

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

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

$$R_i = \frac{h^4}{180} f^{(4)}(x)$$ - 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 $(I - I_1) < \varepsilon$, где I и I_1 - значение интеграла на текущем и предыдущем шаге, соответственно.

4.4. 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 12: **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
  i=0
  FOR i=x1 TO x2
  FOR j=y1 TO y2
    IF POINT (i,j)=1 THEN i=i+1: ax(i)=i-x1:ay(i)=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.

В программе можно предусмотреть демонстрацию возможностей программы.

4.5. 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 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 ролики. Рассмотрим в качестве примера механизм, представленный на рис. 4.1.

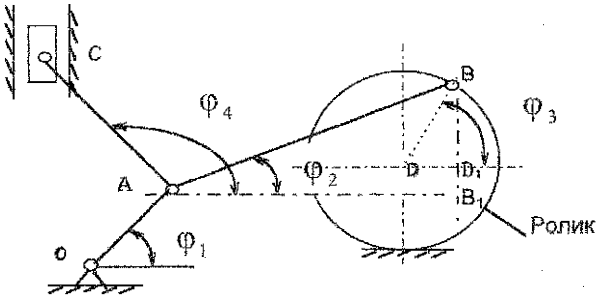


Рис. 4.1. Механизм

На данном рисунке ролик имеет две степени свободы. Он может вращаться вокруг своей оси и перемещаться в горизонтальном направлении. В исходных данных высота расположения ролика и радиус ролика заданы, а следовательно и вертикальная координата центра ролика Y_d известна. Горизонтальное положение ролика 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:

$$\begin{cases} x_b = x_a + l_2 \mathrm{Cos}\varphi_2 & (4.12) \\ x_b = x_d + l_3 \mathrm{Cos}\varphi_3 & (4.13) \\ y_b = y_a + l_2 \mathrm{Sin}\varphi_2 & (4.14) \\ y_b = y_d + l_3 \mathrm{Sin}\varphi_3 & (4.15) \\ (x_b - x_d)^2 + (y_b - y_d)^2 = l_3^2 & (4.16) \end{cases}$$

Здесь l_2 - длина шатуна AB, l_3 - радиус ролика. Первые четыре уравнения получают из рассмотрения прямоугольников ABB_1 и DBD_1. Пятое уравнение составляется из следующих соображений: координаты точки B лежат на поверхности окружности, поэтому можно записать уравнение окружности с центром в точке D и радиусом, равным радиусу ролика l_3.

Из уравнений 4.12 и 4.13 находим значение X_d:

$$x_d = x_a + l_2 \mathrm{Cos}\varphi_2 - l_3 \mathrm{Cos}\varphi_3 \qquad (4.17)$$

Из уравнений 4.14 и 4.15 находим значение угла φ_3:

$$\varphi_3 = \arcsin((y_a + l_2 \mathrm{Sin}\varphi_2 - y_d)/l_3) \qquad (4.18)$$

Заменив в выражении (4.17) угол φ_3 на его значение из (4.18) и подставив значения x_b (4.12), x_d и y_b (4.14) в (4.16) получим уравнение

$$((x_a + l_2 \mathrm{Cos}\varphi_2 - (x_a + l_2 \mathrm{Cos}\varphi_2 - l_3 \mathrm{Cos}(\arcsin((y_a + l_2 \mathrm{Sin}\varphi_2 - y_d)/l_3)))^2 +$$
$$+(y_a + l_2 \mathrm{Sin}\varphi_2 - y_d)^2) - l_3^2 = 0. \qquad (4.19)$$

В выражении 4.19 однин 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.

Пример 13: **Программа для подбора значения угла φ_2**

```
REM  Программа построения механизма
screen 2
window (0,0)-(532,400): REM Переопределение координат на матема-
                        REM тические
REM Ввод исходных данных
pi=atn(1)*4
e=1 : REM 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 m2
    f3=atn(a/sqr(1-a^2))
    xb=xa+l2*cos(f2)
    yb=ya+l2*sin(f2)
```

```
    xd=xb-l3*cos(f3)
  m2: 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
```

5. 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.

Учебное издание

Составитель: Быков Вячеслав Леонидович

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

для выполнения курсовых работ по дисциплинам "Вычислительная техника, программирование и математическое моделирование", "Вычислительная техника и программирование"

Ответственный за выпуск Быков В. Л.
Редактор Строкач Т. В.

Подписано к печати 9.09.98 г. Бумага писчая № 1. Формат $60\times84/_{16}$. Печать офсетная. Усл. печ. л. 1,4. Уч. изд. л. 1,5. Тираж 150 экз. Зак. № 253. Отпечатано на ризографе Брестского политехнического института. 224017, г. Брест, ул. Московская, 267