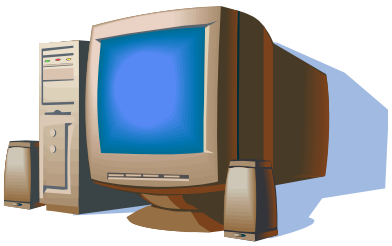
Министерство высшего и среднего специального образования Республики Узбекистан

Наманганский инженерно-педагогический институт

Кафедра «Информатика и информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных работ по дисциплине «Технология программирования»



Наманган-2008

Данные методические указания предназначены для студентов, обучающихся по направлению 5140900 - ПО (Информатика и информационные технологии), и содержат рекомендации по выполнению лабораторных работ и варианты заданий по дисциплине «Технология программирования».

Методические указания могут быть использованы студентами, изучающими дисциплину «Технология программирования», магистрами и преподавателями.

Составители: ст.пр. Жумабаев А.,
асс. Л. Зинина

Обработанные и отредактированные методические указания рассмотрены и одобрены на заседании №___ кафедры «Информатика и информационные технологии» от «___»____200_ г.

Методические указания рассмотрены и одобрены на совете факультета «Информатика и ЭЭ» (Протокол №___ от «___»____200__г.).

Рекомендовано к печати, получив положительный отзыв научно-методического совета Нам ИПИ (Протокол №___ от «___»____200__г.).

Введение

Данные методические указания разработаны для использования при проведении лабораторных занятий по дисциплине «Технология программирования». Они включают в себя краткие теоретические сведения о технологии программирования, программировании на языке Паскаль, варианты заданий для выполнения лабораторной работы, анализ полученных результатов и контрольные вопросы. В конце расположен список использованной литературы. Методические указания могут быть использованы при изучении данной дисциплины студентами, магистрами и преподавателями.

Порядок выполнения лабораторной работы:

1. Сбор теоретических сведений по теме лабораторной работы из указанных источников.
2. Получение индивидуального задания по варианту.
3. Овладение полной информацией для решения конкретной задачи, поставленной в данной лабораторной работе.
4. Описание действий, выполняемых при решении поставленной задачи.
5. Выполнение лабораторной работы на компьютере и сохранение ее в виде графического файла.
6. Печать полученных результатов в соответствии с индивидуальным заданием.
7. Защита выполненной лабораторной работы.
8. Сдача защищенной и оцененной лабораторной работы преподавателю в виде отчета.
9. Студент, не сдавший вовремя лабораторную работу, выполняет и защищает работу на кафедре в указанном порядке.

Примечание: Если студент не выполняет лабораторную работу ни во время, данное преподавателем, ни в дополнительный срок, то он не получает рейтингового балла по данной теме и считается неуспевающим по данной дисциплине.

Лабораторная работа №1

Тема: Проектирование программы. Системный анализ поставленной задачи. Определение этапов создания программного обеспечения**.**

Цель работы: ***Ознакомление студентов с этапами создания программного обеспечения.***

Оборудование: ***учебный компьютер с установленным соответствующим программным обеспечением.***

План:

1. Проектирование программы. Этапы создания программного обеспечения
2. Пример выполнения лабораторной работы
3. Варианты заданий для выполнения лабораторной работы.
4. Контрольные вопросы.

1. Проектирование программы. Этапы создания программного обеспечения

Решение задач с помощью компьютера включает в себя следующие основные этапы, часть из которых осуществляется без участия компьютера.

1. Постановка задачи:
   - сбор информации о задаче;
   - формулировка условия задачи;
   - определение конечных целей решения задачи;
   - определение формы выдачи результатов;
   - описание данных (их типов, диапазонов величин, структуры и т.п.).
2. Анализ и исследование задачи, модели:
   - анализ существующих аналогов;
   - анализ технических и программных средств;
   - разработка математической модели;
   - разработка структур данных.
3. Разработка алгоритма:
   - выбор метода проектирования алгоритма;
   - выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
   - выбор тестов и метода тестирования;
   - проектирование алгоритма.
4. Программирование:
   - выбор языка программирования;
   - уточнение способов организации данных;
   - запись алгоритма на выбранном языке программирования.
5. Тестирование и отладка:
   - синтаксическая отладка;
   - отладка семантики и логической структуры;
   - тестовые расчеты и анализ результатов тестирования;
   - совершенствование программы.
6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2 — 5.
7. Сопровождение программы:

- доработка программы для решения конкретных задач;
- составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

2. Пример выполнения лабораторной работы

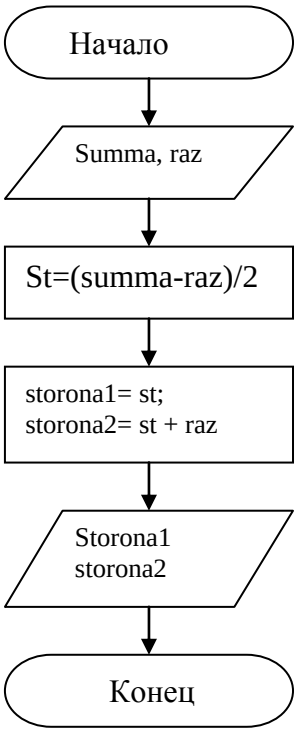
Задание. Выполнить все этапы создания программного обеспечения для решения поставленной задачи.

Одна сторона прямоугольной площадки на 5 м. длиннее другой, а сумма их длин равна 17 м. Найти стороны этой площадки.

1 этап. Сформулируем условие задачи: *Одна сторона прямоугольной площадки на raz длиннее другой, а сумма их длин равна summa.* Конечная цель задачи: **Найти стороны этой площадки.** Результат выводится на экран. Исходные данные имеют тип real.

2 этап. Данная задача довольно типична, поэтому ее решение легко может быть реализовано с помощью программной среды любого алгоритмического языка программирования и компьютера. Математической моделью данной задачи является нахождение сторон прямоугольника с помощью уравнений a-b=5, a+b=17. Из данных формул получаем рабочую формулу st=(summa - raz) / 2, где Для реализации ее решения мы будем использовать язык программирования Паскаль.

3 этап. Воспользуемся алгоритмом линейной структуры.



4 этап. Составляем программу на языке Паскаль, т.к. данный язык является алгоритмическим языком программирования высокого уровня и наиболее соответствует требованиям, необходимым для решения данной задачи:

```
uses crt;
var summa,raz,st:real;
storona1,storona2:real;
begin
ClrScr;
writeln('Введите сумму длин сторон прямоугольника');
readln(summa);
writeln('Введите на сколько одна сторона больше другой');
readln(raz);
st:= (summa - raz) / 2;
storona1:= st;
storona2:= st + raz;
write ('ширина-',storona1:7:2,'см.');
write ('длина-',storona2:7:2,'см.');
end.
```

5 этап. Можно снабдить программу пояснениями, комментариями, однако т.к. составленная программа является довольно простой, то тех комментариев, которые присутствуют в тексте программ, достаточно.

6 этап. После реализации программы на компьютере были введены следующие тестовые исходные данные и получены соответствующие результаты:

Исходные данные		**Результаты**	
Summa	Raz	Storona1	Storona2
15	3	6	9
52	8	22	30
24	0	12	12
30	5	12,5	17,5

Это подтверждает правильность составленной программы

7 этап. Введем исходные данные суммы сторон прямоугольника и их разности: 10 и 4. Тогда значение переменной st будет равно 3, что будет являться одной стороной треугольника (storona1), а переменной storona2 будет присвоено значение 7.

2. Варианты заданий для выполнения лабораторной работы.

1. Нефтебаза отпустила за два дня 2560 л. бензина. Во второй день база отпустила на 280 л. больше. Сколько литров бензина база отпустила отдельно за каждый день?
2. Составьте алгоритм и программу для определения сдачи после покупки в магазине товара: перчаток стоимостью а руб., портфеля стоимостью б руб., галстука стоимостью с руб. Исходная сумма, выделенная на покупку d руб. В случае нехватки денег сдача получится отрицательной.
3. В течении месяца продавец доставлял на дом 4 л молока в день. В марте молоко стоило х руб за литр С первого апреля цена молока увеличилась до [x+a] руб за литр Сколько надо заплатить продавцу за все доставленное молоко в конце апреля? Кол-во покупаемого молока осталось прежним
4. Хозяин хочет оклеить обоями длинную стену в своем доме. Длина этой стены равна а и высота б. Рулон обоев имеет длину 12 м и ширину 1 м. Сколько будут стоить обои для всей стены если цена одного рулона к руб?

5. Некоторый автомат может запросить два числа и выполнить 3 команды. Команда а преобразует имеющуюся пару чисел (x,y) в пару (x-y,y) команда б преобразует пару чисел (x,y) в пару (x+y,y) команда с преобразует пару (ч,y) в (y,x) Составьте алгоритм и программу работы автомата
6. Составьте алгоритм и программу выбирающую из трех чисел то которое лежит между двумя другими
7. Определить какая из двух точек находится дальше от: начала координат; окружности заданного радиуса с центром в начале координат
8. Определите из двух девочек старшую
9. Написать алгоритм читающий "N" и выдающий на экран квадраты чисел от 1 до "N"
10. Вывести номер координаты пункта В наиболее удаленного от пункта А
11. Занести в таблицу оценки учеников класса за год по математике и информатике
12. Напечатать фамилию чемпиона и его результат
13. Сформировать список учащихся сдавших экзамен на отлично
14. Выбрать самого высокого ученика по данным из таблицы
15. Масса 8 литров бензина 5,68 кг. Цистерна имеет объем 500 м^3. Хватит ли ее, чтобы вместить А т бензина?
16. Кусок медного провода длиной 5 м имеет массу 430 г. чтобы провести проводку в квартире требуется С метров. Хватит ли для этой цели мотка провода массой М г?
17. Ракета запускается с точки на экваторе и развивает скорость v км/с. Каков результат запуска? Замечание: если v<=7.8 км/с, то ракета упадет на Землю, если 7.8<v<11.2, то ракета станет спутником Земли, если 11.2≤v≤16.4, то ракета станет спутником Солнца, если v>16.4, то ракета покинет Солнечную Систему.
18. К финалу конкурса лучшего по профессии «Специалист электронного офиса» были допущены трое: Иванов, Петров и Сидоров. Соревнования проходили в три тура. Иванов в первом туре набрал m1 баллов, во втором – n1, а в третьем – p1. Петров - соответственно m2, n2, p2; Сидоров – m3, n3, p3 баллов. Составьте программу, определяющую: а) сколько баллов набрал победитель; б) фамилию победителя.
19. Найдите наибольший общий делитель двух заданных целых чисел.
20. Найдите наименьшее общее кратное двух заданных целых чисел.
21. Определите, является ли заданное число нечетным двузначным числом.
22. Заданы площади квадрата и круга. Определите, поместится ли квадрат в круге.
23. Определите, имеют ли общие точки две плоские фигуры — треугольник с заданными координатами его вершин и круг заданного радиуса с центром в начале координат.
24. Задано целое A > 1. Найдите наименьшее целое неотрицательное k, при котором 2k > A.
25. Преобразуйте число, заданное в римской системе счисления, в число десятичной системы.

4. Контрольные вопросы.

1. Какие основные этапы включает в себя решение задач на компьютере?
2. Какие этапы компьютерного решения задач осуществляются без участия компьютера?
3. Что называют математической моделью объекта или явления?
4. Почему невозможно точное исследование поведения объектов или явлений?
5. Какие способы моделирования осуществляются с помощью компьютера?
6. Из каких последовательных действий состоит процесс разработки программы?

Лабораторная работа №2

Тема: ***Разработка структуры данных. Создание алгоритма для поставленной задачи. Словесное описание алгоритма. Алгоритм в виде блок-схемы***

Цель работы: ***Ознакомление студентов с этапами создания программного обеспечения.***

Оборудование: ***учебный компьютер с установленным соответствующим программным обеспечением.***

План:

1. Структура данных.
2. Создание алгоритма для поставленной задачи. Способы описания алгоритма.
3. Примеры выполнения лабораторной работы.
4. Варианты заданий для выполнения лабораторной работы.
5. Контрольные вопросы.

1. Структура данных

Исходные данные должны быть полными, т.е. содержать данные, необходимые и достаточные для решения задачи. Если данные неполные, необходимо приложить дополнительные усилия для сбора дополнительных сведений; эта ситуация может также возникнуть на последующих этапах при выборе метода решения.

Различают исходные данные трех видов: постоянные, условно-постоянные и переменные.

Постоянные исходные данные - это данные, которые сохраняют свои значения в процессе решения задачи (математические константы, координаты неподвижных объектов) и не зависят от внешних факторов.

Условно-постоянные данные - это данные, которые могут иногда изменять свои значения; причем эти изменения не зависят от процесса решения задачи, а определяются внешними факторами (величина подоходного налога, курс валют, количество дней в году).

Переменные данные - это данные, которые изменяют свои значения в процессе решения задачи.

На этом этапе важно не только классифицировать данные по отношению к процессу решения, но определить их наименование, тип, структуру и ограничения, накладываемые на значения. Желательно также определить допустимые и недопустимые операции по отношению к различным типам исходных данных.

Классификация данных по структурному признаку

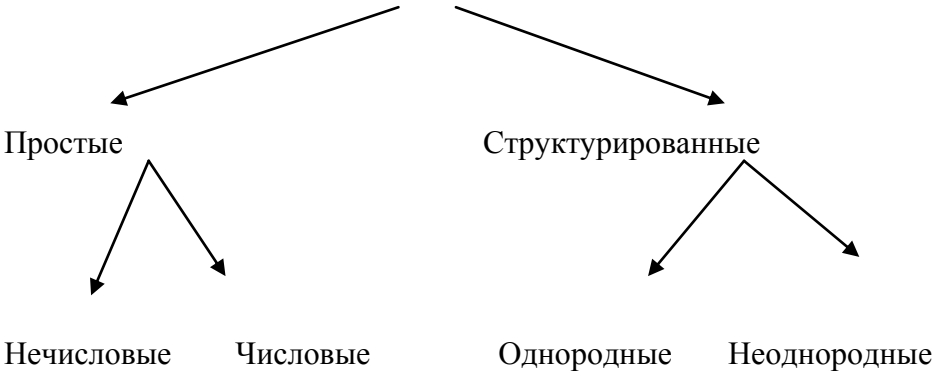


Рис.1.

Данное относят к простому типу, если в любой момент времени оно определяет одно и только одно значение. Диапазон изменения возможных значений определяется типом данных. Например, требуется вычислить площадь поверхности некоторого тела. Очевидно, что для представления информации о вычисляемой площади поверхности некоторого тела достаточно использовать данное простого числового типа. Простые данные определяют такое отношение: одно имя - одно значение.

Структурированные данные отличаются от простых тем, что к ним применимо другое отношение: одно имя - много значений. Если все элементы, входящие в такую структуру, однотипны, то такая структура называется однородной, в противном случае - неоднородной. Классическим примером однородной структуры является массив, являющийся последовательностью однотипных значений, таких как, например, (2,51,3,7,88). Неоднородная структура в отличие от однородной содержит значения различных типов, относящихся к одному понятию или объекту, и, значит, такое структурированное данное несет в себе больше информации. Для представления неоднородных структур используют запись. Запись - это структура, предназначенная для представления данных различного типа. Запись состоит из поименованных полей, каждое из которых должно содержать значение определенного типа.

Рассмотрим простой пример. Задача заключается в определении в некоторой стране города с максимальным количеством жителей. Данные, которые необходимо проанализировать, это нечисловые данные, содержащие информацию о названии города, и числовые данные, содержащие информацию о численности населения в этом городе. В качестве структуры, содержащей данные о названии города и количестве в нем жителей, следует выбрать неоднородную структуру - запись, пример которой изображен в таблице 1.

Таблица 1.Пример записи

Имя поля: *Город*	Имя поля: *Количество жителей*
Тип поля: *Строка символов*	Тип поля: *Число*
Значение: *Москва*	Значение: *8 578 676*

В качестве структуры, содержащей информацию о множестве городов рассматриваемой страны, можно выбрать однородную структуру типа массив, состоящий из записей таблицы 1.

Определение отношений между данными, условий и ограничений, накладываемых на значения данных и эти отношения, зависит от конкретной постановки задачи и требований пользователя.

В результате анализа постановка и требования задачи могут быть представлены в обобщенном виде.

2. Создание алгоритма для поставленной задачи. Способы описания алгоритма.

Алгоритм моделирует решение задачи в виде точно определенной последовательности действий для некоторого исполнителя по преобразованию исходных данных в результирующие. Процесс составления алгоритмов называют алгоритмизацией.

Алгоритм, реализующий решение задачи, можно представить различными способами: с помощью графического или текстового
описания, в виде таблицы значений. Графический способ представления алгоритмов имеет ряд преимуществ благодаря визуальности и явному отображению процесса решения задачи. Алгоритмы, представленные графическими средствами, получили название визуальные алгоритмы. Текстовое описание алгоритма является достаточно компактным и может быть реализовано на абстрактном или реальном языке программирования в виде программы для ЭВМ. Таблицы значений представляют алгоритм неявно, как некоторое преобразование конкретных исходных данных в выходные. Табличный способ описания алгоритмов может быть с успехом применен для проверки правильности функционирования разработанного алгоритма на конкретных тестовых наборах входных данных, которые вместе с результатами выполнения алгоритма фиксируются в "таблицах трассировки".

Таким образом, все три способа представления алгоритмов можно считать взаимодополняющими друг друга. На этапе проектирования алгоритмов наилучшим способом является графическое представление, на этапе проверки алгоритма - табличное описание, на этапе применения - текстовая запись в виде программы.

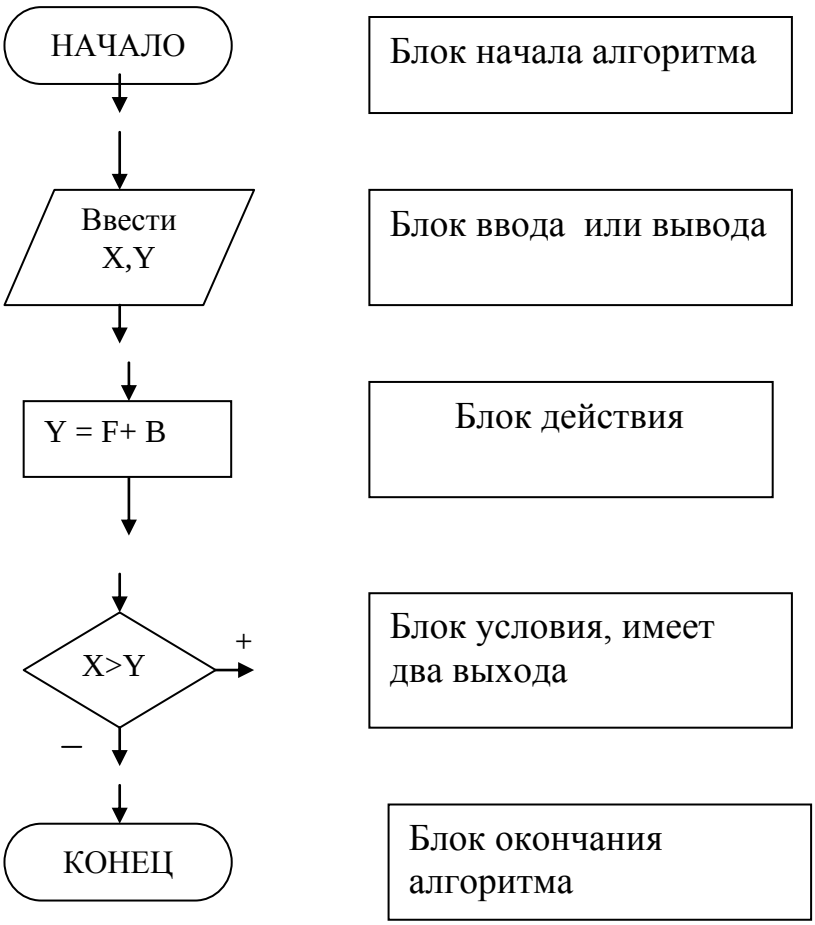
При проектировании визуальных алгоритмов используют специальные графические элементы, называемые графическими блоками. Результатом алгоритмизации решения задачи является блок-схема алгоритма, состоящая из некоторой последовательности таких графических блоков.

Общими правилами при проектировании визуальных алгоритмов являются следующие:

- В начале алгоритма должны быть блоки ввода значений входных данных.
- После ввода значений входных данных могут следовать блоки обработки и блоки условия.
- В конце алгоритма должны располагаться блоки вывода значений выходных данных.
- В алгоритме должен быть только один блок начала и один блок окончания.
- Связи между блоками указываются направленными или ненаправленными линиями.

Этап проектирования алгоритма следует за этапом формального решения задачи, на котором определены входные и выходные данные, а также зависимости между ними.

При построении алгоритмов для сложной задачи используют системный подход с использованием декомпозиции (нисходящее проектирование сверху-



вниз). Как и при разработке любой сложной системы, при построении алгоритма используют дедуктивный и индуктивный методы. При дедуктивном методе рассматривается частный случай общеизвестных алгоритмов. Индуктивный метод применяют в случае, когда не существует общих алгоритмических решений.

3. Примеры выполнения лабораторной работы.

1. Пример линейного алгоритма приведен на рис 3.

2.Составить алгоритм нахождения минимального значения из 3-х чисел.

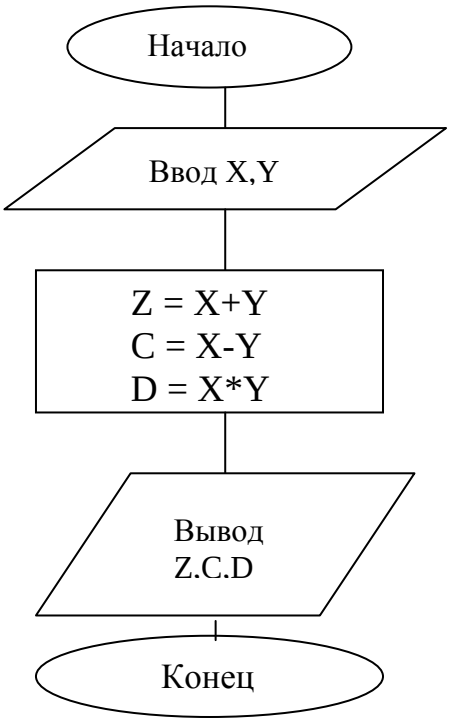


Рис. 3.

Решение. Для определения минимального значения будем использовать проверку пары значений. Визуальный разветвленный алгоритм приведен на рис.4.

3. Составить алгоритм определения находится ли точка М с координатами Х,У на окружности радиуса R. *Решение.* Визуальный алгоритм приведен на рис. 5. Для решения в нем используется математическая модель в виде формулы окружности $R^2 = X^2+Y^2$.

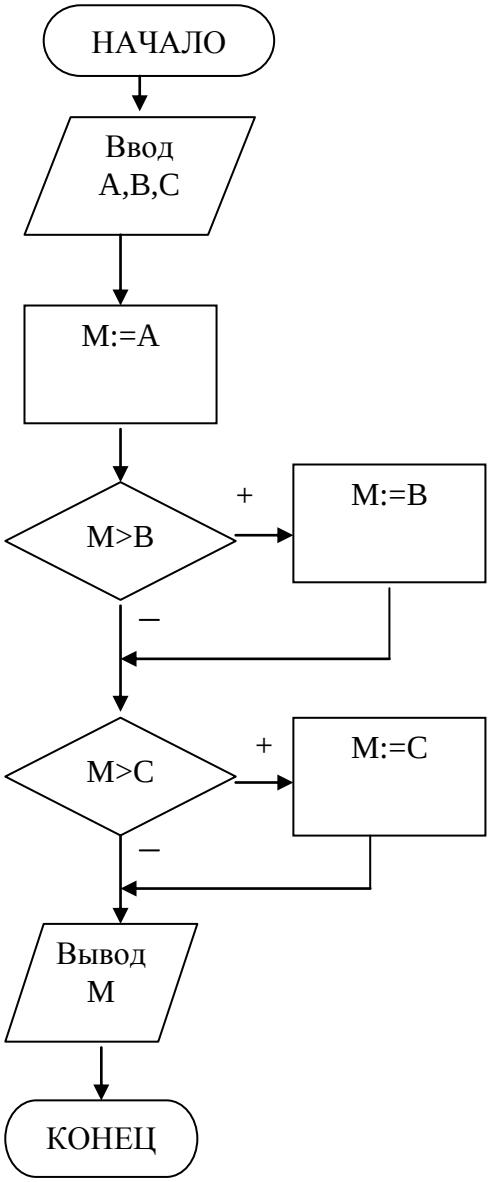


Рис. 4.Поиск минимального числа из трёх A,B,C. Метод сравнения с промежуточной переменной M.

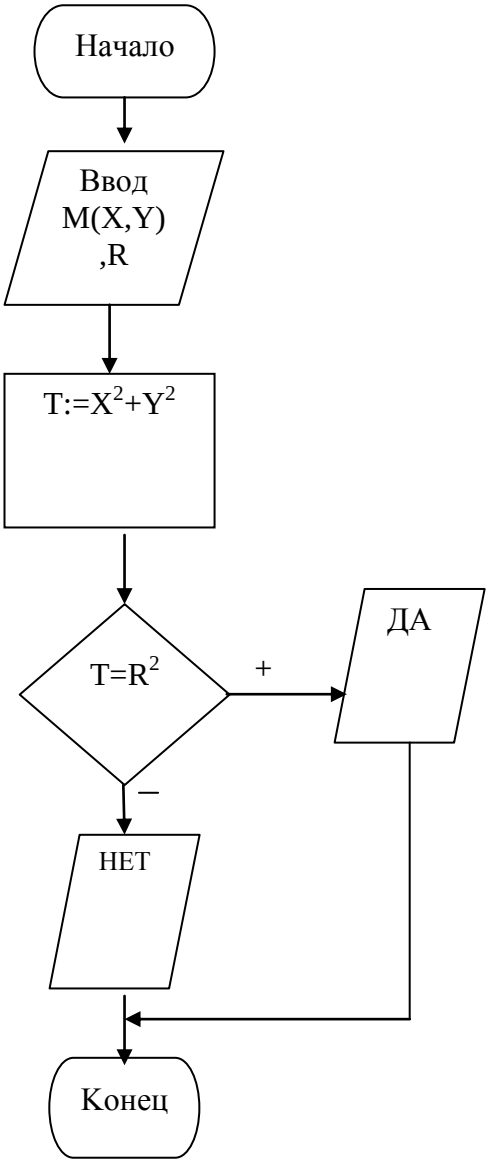


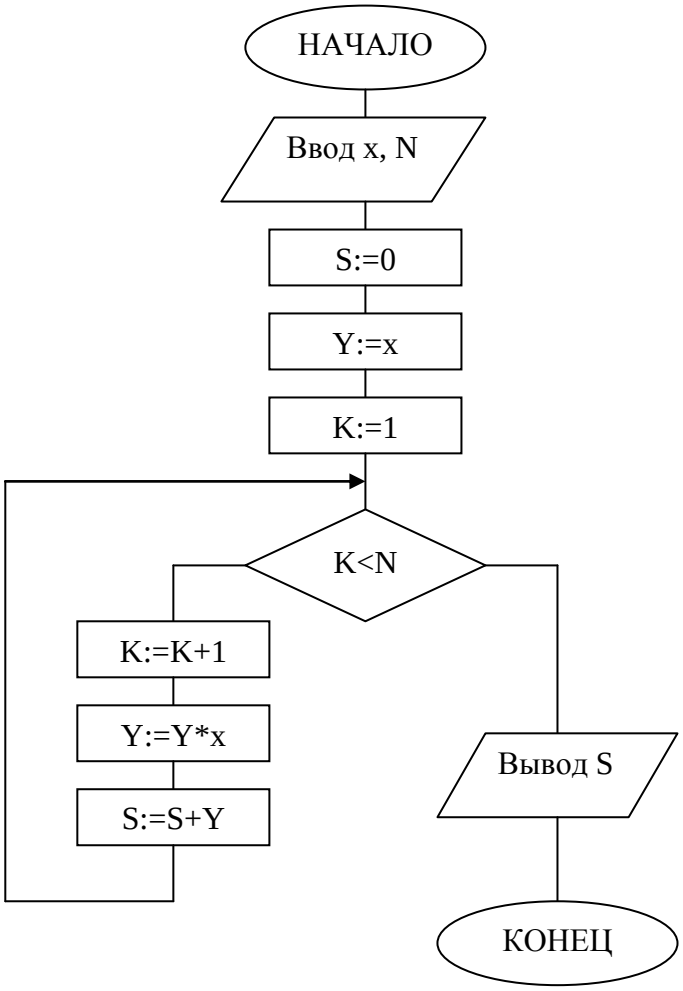
Рис. 5. Определить находится ли точка M с координатами X,У на окружности радиуса R.

3. Пусть требуется составить алгоритм вычисления суммы ряда S=x+x^2+x^3+…+x^n.

Решение. Исходные данные для алгоритма это переменные x и n. На каждом шаге будем вычислять очередной член суммы Y и прибавлять его к предыдущему значению суммы S.Для этого используем реккурентную формулу вычисления степени X (см. таблицу 2) Y=Y*X, тогда сумма ряда на каждом шаге итерации будет вычисляться по формуле S=S+Y. Количество итераций K изменяется от 1 до n и равно количеству членов ряда. Начальное значение суммы ряда S равно 0. На рис. 6 представлен циклический алгоритм с предусловием для вычисления заданной суммы ряда.

Таблица 2.Реккурентные соотношения при вычислении $Y=X^n$

N	Y	Реккурентные соотношения
1	Y[1]=X	Y=X
2	Y[2]=X*X или Y[2]=Y[1]*X	Y=X*X или Y=Y*X
3	Y[3]=X*X*X или Y[3]=Y[2]*X	Y=X*X*X или Y=Y*X



4. Варианты заданий для выполнения лабораторной работы.

1. Ввести массив a1,a2,...,a15. Требуется упорядочить его по возрастанию абсолютных значений элементов
2. Ввести массив x1,x2,...,x20. Требуется расположить отрицательные элементы в порядке убывания.
3. В одномерном массиве определить первый отрицательный элемент и его номер.
4. Исключить из массива A1..AN первый отрицательный элемент.

5. Исключить из массива A1..AN первый четный элемент, следующий за максимальным.
6. Дан массив целых чисел a1,..an.Выяснить, какая из трех ситуаций имеет место:все числа a1,..an равны нулю, в последовательности a1,...,an первое ненулевое число-положительное, первое ненулевое число-отрицательное.
7. Дан массив целых чисел a1,..an.Выяснить, какая из трех ситуаций имеет место:все числа a1,..an равны нулю, в последовательности a1,...,an первое ненулевое число-положительное,первое ненулевое число-отрицательное.
8. Даны целые числа a1,..,an.Определить количество целых чисел, входящих в последовательность a1,...,an по одному разу.
9. Даны действительные числа a1,..,an.Требуется найти B равное среднему арифметическому чисел a1,..,an,и наибольшее отклонение от среднего,т.е. max(/a1-b/,/a2-b/,../an-b/).
10. Дан массив действительных чисел a1,...,an.Найти максимальный элемент среди отрицательных элементов и поменять его местами с минимальным положительным.
11. В одномерном массиве перенести в начало максимальный элемент.
12. Перенести в начало одномерного массива второй нулевой элемент.
13. Ввести массив a1,...,a16. Получить новый массив по правилу (a1 + a16, a2+a15,...,a8+a9). Найти минимальный элемент полученного массива.
14. В одномерном массиве перенести в конец минимальный элемент .
15. Перенести в хвост одномерного массива все отрицательные элементы.
16. Перенести в начало одномерного массива все нечетные элементы.
17. В одномерном массиве найти первую группу повторяющихся элементов.
18. Выполните примеры 10 и 11, реализуя ввод элементов массива в цикле, в котором производится их обработка.
19. Ввести упорядоченный по неубыванию массив X(1) < = X(2) < =…X(n). Найти количество различных чисел среди элементов этого массива.
20. Ввести два упорядоченных по возрастанию числовых массива X(1)<X(2)<X(3) <…X(n) и Y(1)<Y(2)<…Y(m). Найти количество общих элементов в этих массивах, то есть количество K таких чисел X(i)= Y(j).
21. Известно, что некоторое число содержится в каждом из трех целочисленных неубывающих массивов X(1) < = X(2) < =…X(n), Y(1)< =Y(2)< =…Y(m) и Z(1) < = Z(2) < = … Z(k).Найти одно из этих чисел.
22. Вставить значение P в упорядоченный неубыванию массив X(1) < = X(2) < =…X(n) так, чтобы упорядоченность не нарушилась.
23. Удалить значение P в упорядоченный неубыванию массиве X(1) < = X(2) < =…X(n).
24. Соединить два упорядоченных массива X(1) < = X(2) < =…X(n) и Y(1)< =Y(2)< =…Y(m) в массив Z(1) < = Z(2) < = … Z(k), при этом каждый элемент должен входить в массив Z столько раз, сколько раз он входит в массивы X и Y.
25. Ввести массив a1,a2,...,a15. Расположить ненулевые элементы по убыванию.

5. Контрольные вопросы.

1. По каким признакам разделяют данные?
2. Что такое алгоритм?
3. Что такое алгоритм линейной структуры?
4. Что такое алгоритм разветвляющейся структуры?
5. Что такое алгоритм циклической структуры?

Лабораторная работа №3

Тема: ***Создание программы. Создание программы с помощью языков программирования. Оценка погрешностей программы и отладка. Анализ полученных результатов. Связь полученного результата с постановкой задачи. Процедуры и функции. Модули.***

Цель работы: ***Ознакомление студентов с этапами создания программного обеспечения.***

Оборудование: ***учебный компьютер с установленным соответствующим программным обеспечением.***

План:

1. Создание программы с помощью языков программирования.
2. Оценка погрешностей программы и отладка.
3. Анализ полученных результатов.
4. Процедуры и функции. Модули.
5. Варианты заданий для выполнения лабораторной работы.
6. Контрольные вопросы.

1. Создание программы с помощью языков программирования.

При разработке программного модуля рекомендуется придерживаться следующего порядка:

– изучение и проверка спецификации модуля, выбор языка программирования;
– выбор алгоритма и определение структуры данных;
– программирование (кодирование) модуля;
– шлифовка или доводка текста модуля;
– проверка модуля;
– компиляция модуля.

Первый этап разработки программного модуля во многом является смежным контролем структуры программы снизу. Изучение спецификации модуля необходимо разработчику для проектирования этого модуля. После чего выбирается язык программирования, который может быть либо уже предопределён для всего ПС, либо может быть выбран другой язык, более подходящий для реализации данного модуля (например, язык ассемблера).

На втором этапе разработки программы необходимо определить, есть ли уже какие-либо алгоритмы для решения поставленной задачи. И если найдётся подходящий алгоритм, то целесообразно им воспользоваться. Определение подходящих структур данных во многом предопределяет логику и качественные характеристики разрабатываемого модуля.

На третьем этапе создаётся текст модуля на выбранном языке программирования. Необходимость учёта всевозможных деталей при реализации функций, указанных в спецификации модуля, легко может привести к созданию очень сложного и запутанного текста. Поиск ошибки в таком «непрозрачном» модуле и внесение в него изменений может оказаться трудоёмкой задачей. ***Поэтому для построения текста модуля важно пользоваться обоснованной технологически и проверенной на практике дисциплиной программирования.*** Первым об этом заговорил ***Дейкстра,*** сформулировав и обосновав основные ***принципы структурного программирования***. Эти принципы лежат в основе многих дисциплин программирования. Наиболее распространена ***дисциплина пошаговой детализации***.

На четвёртом этапе разработки ПМ на основе спецификации качества ПС текст модуля доводится до завершённого состояния. На третьем этапе основное внимание уделяется правильности реализации функций модуля. При доводке текста модуля разработчик редактирует имеющиеся комментарии и, возможно, включает дополнительные для обеспечения требуемых примитивов качества.

Пятый этап является ручной проверкой внутренней логики ПМ до начала его отладки («прогоны» его на компьютере).

На шестом этапе разработки завершается проверка модуля (с помощью компилятора) и становится возможным переход к процессу отладки модуля.

2. Оценка погрешностей программы и отладка.

Отладка программного средства – это деятельность, в результате которой обнаруживаются и исправляются ошибки программного средства в процессе выполнения его программ.

Тестирование программного средства – процесс, при котором выполняются программы программного средства на некотором наборе исходных данных. Причём заранее известен результат выполнения этих программ. Указанный набор данных называется тестовым набором (тестом). Таким образом, процесс отладки можно представить в виде многократного повторения трёх операций:

– тестирование, которое позволяет определить в программном средстве наличие ошибки;

– локализация ошибки в программах и документации программного средства;

– редактирование программ и документации программного средства для устранения обнаруженных ошибок.

Автономная отладка программного средства – последовательное раздельное тестирование различных частей программ программного средства с поиском и исправлением в них обнаруженных при тестировании ошибок. Автономная отладка программного средства включает отладку каждого программного модуля и отладку сопряжения модулей.

Комплексная отладка программного средства – тестирование программного средства в целом. Представляет собой поиск, исправление фиксируемых при тестировании ошибок во всех документах, относящихся к программному средству.

3. Анализ полученных результатов.

Тестовые данные должны обеспечить проверку всех возможных условий возникновения ошибок:

- должна быть испытана каждая ветвь алгоритма;
- очередной тестовый прогон должен контролировать нечто такое, что еще не было проверено на предыдущих прогонах;
- первый тест должен быть максимально прост, чтобы проверить, работает ли программа вообще;
- арифметические операции в тестах должны предельно упрощаться для уменьшения объема вычислений;
- количества элементов последовательностей, точность для итерационных вычислений, количество проходов цикла в тестовых примерах должны задаваться из соображений сокращения объема вычислений;
- минимизация вычислений не должна снижать надежности контроля;
- тестирование должно быть целенаправленным и систематизированным, так как случайный выбор исходных данных привел бы к трудностям в определении ручным способом ожидаемых результатов; кроме того, при случайном выборе тестовых данных могут оказаться непроверенными многие ситуации;
- усложнение тестовых данных должно происходить постепенно.

Пример. Система тестов для задачи нахождения корней квадратного уравнения ax2 + bx + c = 0 :

Номер теста	**Проверяемый случай**	**Коэффициенты**			**Результаты**
		a	**b**	**c**	
1	**d >0**	**1**	**1**	**-2**	**x1 = 1, x2 = - 2**
2	**d=0**	**1**	**2**	**1**	**Корни равны: x1 = - 1, x2 = - 1**
3	**d < 0**	**2**	**1**	**2**	**Действительных корней нет**
4	**a=0, b=0, c=0**	**0**	**0**	**0**	**Все коэффициенты равны нулю. x — любое число.**
5	**a=0, b=0, c<>0**	**0**	**0**	**2**	**Неправильное уравнение**
6	**a=0, b<>0**	**0**	**2**	**1**	**Линейное уравнение. Один корень: x = - 0,5**
7	**a <> 0, b <> 0, c = 0**	**2**	**1**	**0**	**x1 = 0, x2 = - 0,5**

4. Процедуры и функции. Модули.

Алгоритм решения задачи проектируется путем декомпозиции всей задачи в отдельные подзадачи. Обычно подзадачи реализуются в виде подпрограмм.

Подпрограмма - это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы. Каждая подпрограмма определяется уникальным именем. В языке ПАСКАЛЬ существуют два типа подпрограмм - процедуры и функции.

Процедура и функция - это именованная последовательность описаний и операторов. При использовании процедур или функций ПАСКАЛЬ - программа должна содержать текст процедуры или функции и обращение к процедуре или функции. Тексты процедур и функций помещаются в раздел описаний процедур и функций.

Процедура может содержать такие - же разделы описаний, что и ПАСКАЛЬ - программа, а именно: разделы описания модулей, меток, констант, типов, переменных, процедур и функций.

Во многих задачах, особенно в задачах вычислительной математики, необходимо передавать имена процедур и функций в качестве параметров. Для этого в TURBO PASCAL введен новый тип данных - процедурный или функциональный, в зависимости от того, что описывается.

Описание процедурных и функциональных типов производится в разделе описания типов:

```
type
    FuncType = Function(z: Real): Real;
    ProcType = Procedure (a,b: Real; var x,y: Real);
```

Функциональный и процедурный тип определяется как заголовок процедуры и функции со списком формальных параметров, но без имени. Можно определить функциональный или процедурный тип без параметров, например:

```
type
    Proc = Procedure;
```

После объявления процедурного или функционального типа его можно использовать для описания формальных параметров - имен процедур и функций.

Кроме того, необходимо написать те реальные процедуры или функции, имена которых будут передаваться как фактические параметры. Эти процедуры и функции должны компилироваться в режиме дальней адресации с ключом {$F+}.

Пример. Составить программу для вычисления определенного интеграла

$$I = \int_{tn}^{tk} \frac{2t}{\sqrt{1-\sin 2t}}\,dt$$

по методу Симпсона. Вычисление подинтегральной функции реализовать с помощью функции, имя которой передается как параметр. Значение определенного интеграла по формуле Симпсона вычисляется по формуле:

ISimps=2*h/3*(0.5*F(A)+2*F(A+h)+F(A+2*h)+2*F(A+3*h)+...
+2*F(B-h)+0.5*F(B))

где A и B - нижняя и верхняя границы интервала интегрирования, N - число разбиений интервала интегрирования, h=(B-A)/N, причем N должно быть четным.

```
Program INTEGRAL;
 type
   Func= function(x: Real): Real;
 var
   I,TN,TK:Real;
   N:Integer;
{$F+}
 Function Q(t: Real): Real;
  begin
    Q:=2*t/Sqrt(1-Sin(2*t));
  end;
{$F-}
 Procedure Simps(F:Func; a,b:Real; N:Integer; var INT:Real);
  var
     sum, h: Real;
     j:Integer;
  begin
    if Odd(N) then N:=N+1;
    h:=(b-a)/N;
    sum:=0.5*(F(a)+F(b));
    for j:=1 to N-1 do
      sum:=sum+(j mod 2+1)*F(a+j*h);
      INT:=2*h*sum/3
  end;
 begin
  WriteLn(' ВВЕДИ TN,TK,N');
  Read(TN,TK,N);
  Simps(Q,TN,TK,N,I);
  WriteLn('I=',I:8:3)
 end.
```

При разработке каждой программы нужно иметь в виду, что она является большой системой, и мы должны принять меры для её упрощения. Для этого программное средство разрабатывают по частям, которые называются программными модулями. Метод такой разработки программ называют модульным программированием. Программный модуль – это любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях процесса. При этом каждый программный модуль программируется, компилируется и отлаживается отдельно от других модулей программы (физически разделён с другими модулями программы). Кроме того, каждый разработанный программный модуль может

включаться в состав разных программ. Таким образом, программный модуль может рассматриваться и как средство борьбы со сложностью программ, и как средство борьбы с дублированием в программировании.

Применение модульного программирования в процессе разработки программ реализует общие методы борьбы со сложностью, а именно, обеспечение независимости компонент системы и использование иерархических структур

5. Варианты заданий для выполнения лабораторной работы.

1. Найти длину вектора x = {x1, x2, x3}, компонентами которого являются суммы элементов квадратных матриц соответственно A(4, 4), B(3, 3), C(2, 2). Определение суммы элементов квадратной матрицы оформить в виде функции.
2. Определить количество положительных, отрицательных и нулевых элементов в трех заданных матрицах: A(8, 10), B(4, 8), C(3, 7). Подсчет указанных чисел в матрице оформить в виде функций.
3. Найти сумму элементов трех одномерных массивов A(10), B(20), C(30). Вычисление суммы элементов одномерного массива оформить в виде функции.
4. Даны действительные числа a0,…, a6. Получить для x =1, 2, 3, 4 значения P(x + 1) – P(x), где P(y) = a6y6 + a5y5 + … + a0. Вычисление P(y) оформить в виде функции.
5. Определить полусумму длин двух векторов: A(1,5; 2,5; –0,3) и B(-11,7; 9,3; 2,5; 3,7; –1,2). Вычисление длины вектора оформить в виде функции.
6. Вычислить: v = a*b*c*sin(a-c)*exp(–b), где a,b,c – наименьшие элементы двумерных массивов чисел A(10, 10), B(6, 8), C(4, 10). Поиск наименьшего элемента оформить в виде функции.
7. Определить произведение всех элементов матриц A(3, 4), B(6, 8), отличных от нуля. Вычисление оформить в виде функции.
8. Даны действительные числа x1, y1, x2, y2,…, x10, y10. Найти периметр десятиугольника, вершины которого имеют соответственно координаты (x1, y1), (x2, y2),…, (x10, y10).(Оформить в виде функции задачу вычисления расстояния между двумя точками, заданными своими координатами).
9. Даны действительные числа s, t. Вычислить: f(t, -2s, 1,07) + f(2,2, t, s–t), где f (a, b, c) = (2a – b – sin(c)) / (5 + |c|). Вычисление f (a, b, c) оформить в виде функции.
10. Вычислить и записать в массив U(3) большие корни квадратных уравнений ax2 + bx – 4 = 0; y2 – cy + d = 0; 2z2 + dz – 1 = 0. Если корень комплексный, считать его равным нулю. Корни уравнений вычислить в виде функции.
11. Вычислить: S = N! + K! + (I + K)!, где N = 3, K = 5, I = 8. Вычисление факториала оформить в виде функции.
12. Выполнить слияние двух заданных массивов чисел A(10) и B(20) в такой массив, в котором положительные элементы расположились бы в начальной, а отрицательные члены – в концевой части; нулевые члены были бы исключены. Для сортировки массива применить подпрограмму.
13. В целочисленном массиве X(40, 35) определить номер строки, содержащей максимальное число четных элементов. Подсчет числа четных элементов оформить в виде функции.

14. Вычислить средний балл групп по результатам сессии. Оценки групп сведены в матрицы A(25, 5), B(23, 5), C(22, 5), D(24, 5). Средний балл группы оформить в виде функции.
15. В массиве T(33, 19) найти произведение строк: первой на последнюю, второй на предпоследнюю. Вычисление произведения двух строк матрицы оформить в виде функции.
16. В целочисленном массиве Z(10, 15) определить номер столбца содержащей максимальное число нечетных элементов. Подсчет числа нечетных элементов оформить в виде функции.
17. Даны натуральные числа n, m (n<m). Определить, какие из чисел n, n + 1, …, m являются номерами високосных годов. Проверку, является ли год високосным оформить в виде функции.
18. В массиве A(12, 15) найти произведение столбцов: второго на десятый, третьего на девятый. Вычисление произведения двух столбцов матрицы оформить в виде функции.
19. Даны действительные числа a и b. Получить U = min (a, b), V = min (ab, a+b), min (U+V2, 3.14). Вычисление минимального значения двух действительных чисел оформить в виде функции.
20. В трех целочисленных массивах A(40), B(30), C(60) найти все элементы, кратные 3. Поиск элементов однородного массива, кратных некоторому числу P, оформить в виде функции.
21. Вычислить Z = (x + y) / (kn), где x и k – сумма и количество положительных элементов массива A(40), где y и n – сумма и количество отрицательных элементов массива B(50). Определение суммы и количества положительных и отрицательных элементов выполнить в подпрограмме.
22. Определить произведение элементов, отличных от нуля, каждого столбца матрицы A(5, 7). Вычисление произведения элементов матрицы оформить в виде функции.
23. Даны две квадратные матрицы A(3, 3) и B(4, 4). Составьте программу для транспонирования матрицы.
24. Вычислите среднее арифметическое положительных элементов массив X(60), Y(75), Z(80). Определение среднего арифметического оформить в виде функций.
25. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая проверяет их сумму, третья выводи результат. Используйте эти процедуры в основной программе. Используйте X и Y как глобальные переменные.

6. Контрольные вопросы.

1. Что такое тестирование и отладка?
2. Какие существуют виды ошибок?
3. Для чего служат процедуры и функции?
4. Чем модуль отличается от программы?

Лабораторная работа №4

Тема: Создание управляющей программы. Организация пользовательского интерфейса.

Цель работы: ***Ознакомление студентов с этапами создания программного обеспечения.***

Оборудование: ***учебный компьютер с установленным соответствующим программным обеспечением.***

План:

1. Создание управляющей программы.
2. Организация пользовательского интерфейса.
3. Варианты заданий для выполнения лабораторной работы.
4. Контрольные вопросы.

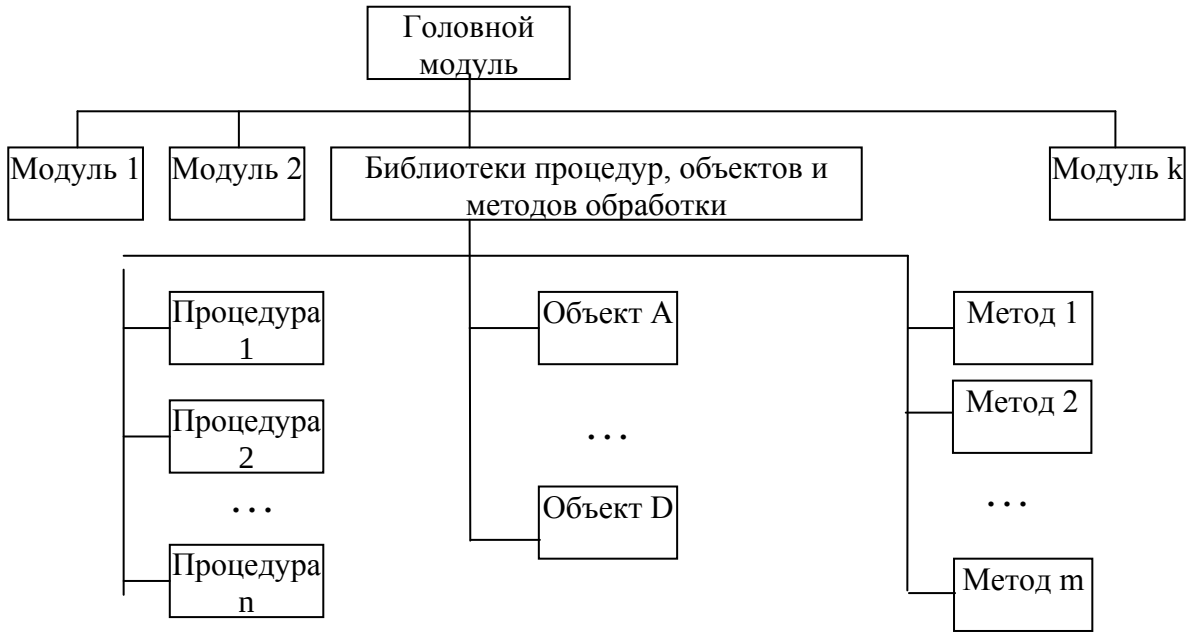
1. Создание управляющей программы.

Программный модуль – это любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях процесса. Это означает, что каждый программный модуль программируется, компилируется и отлаживается отдельно от других модулей программы, и тем самым, физически разделен с другими модулями программы.

Модульное программирование является воплощением в процессе разработки программ общих методов борьбы со сложностью и обеспечение независимости компонент системы, и использование иерархических структур.

Некоторые программные продукты используют модули из готовых библиотек стандартных подпрограмм, процедур, функций, объектов, методов обработки данных

Структура программного продукта



Среди множества модулей различают:

- головной модуль — управляет запуском программного продукта (существует в единственном числе);
- управляющий модуль — обеспечивает вызов других модулей на обработку
- рабочие модули — выполняют функции обработки;
- сервисные модули и библиотеки, утилиты — осуществляют обслуживающие функции.

В работе программного продукта активизируются необходимые программные модули. Управляющие модули задают последовательность вызова на выполнение очередного модуля. Информационная связь модулей обеспечивается за счет использования общей базы данных либо межмодульной передачи данных через переменные обмена.

Пример. Напишем модуль, содержащий процедуру сортировки массива вещественных чисел, и функцию, определяющую сумму его элементов.

```
Unit Sort;
  Interface { интерфейсная секция  }
    Const   {  Глобальная константа и тип }
         Nmax=200;
    Type
         mas = array[1..Nmax] of real;
    Procedure SortY(Var Y : mas; m : integer);
    Function SumMas (Y : mas; m : integer): real;

  Implementation { секция  реализации }
    Procedure SortY;  { можно только имя }
       Var
          i,k : integer;
          a : real;
       Begin
         for k := 1 to m-1 do
           for i := 1 to m-k do
              if Y[i]>Y[i+1]then
                 Begin
                   a := Y[i];
                   Y[i] := Y[i+1];
                   Y[i+1] := a;
                 end;
      end; { SortY }
    Function SumMas;  { можно только имя }
     {Нахождение суммы элементов массива вещественных чисел}
     Var
        I : Integer;
        S : Real; { Вспомогательная переменная - сумма}
     Begin
       S := 0;
       For i := 1 to k Do
             S:=S + X[i];
       SumMas:=S;
     End; { SumMas }
```

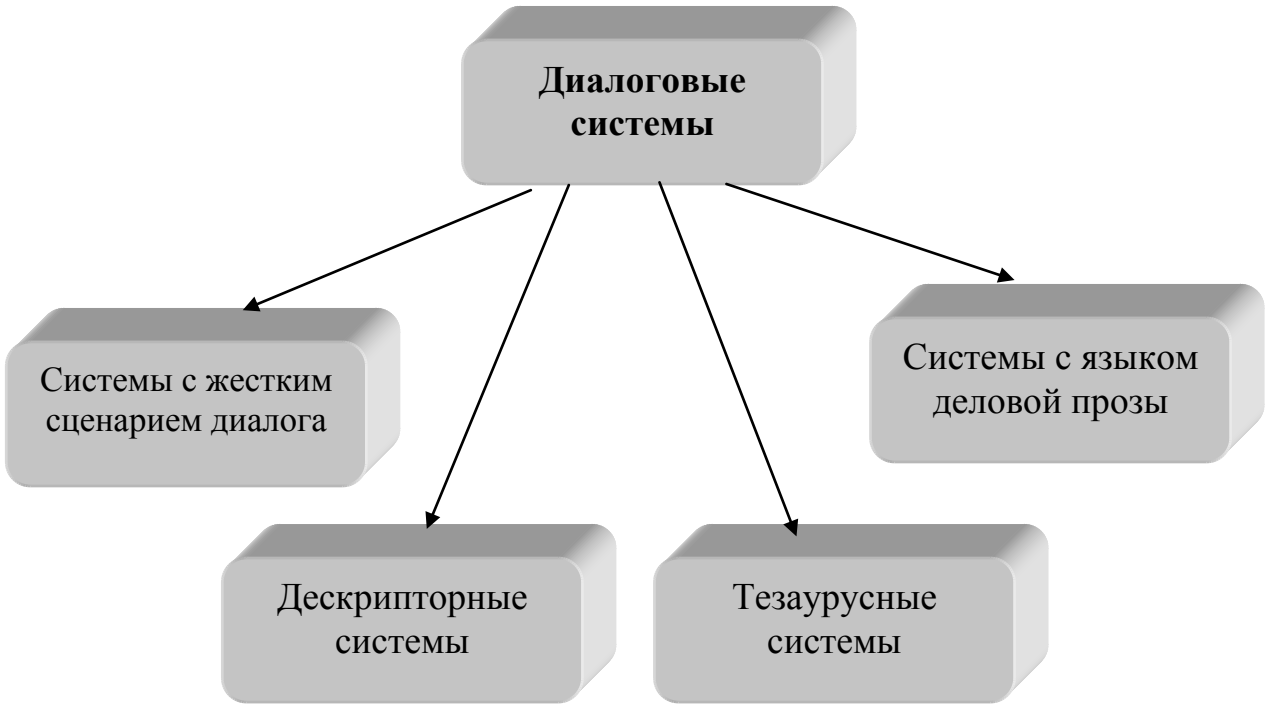
```
Begin
          { секция инициализации не содержит операторов }
End.
```

Основная программа, которая использует этот модуль, может быть такой.

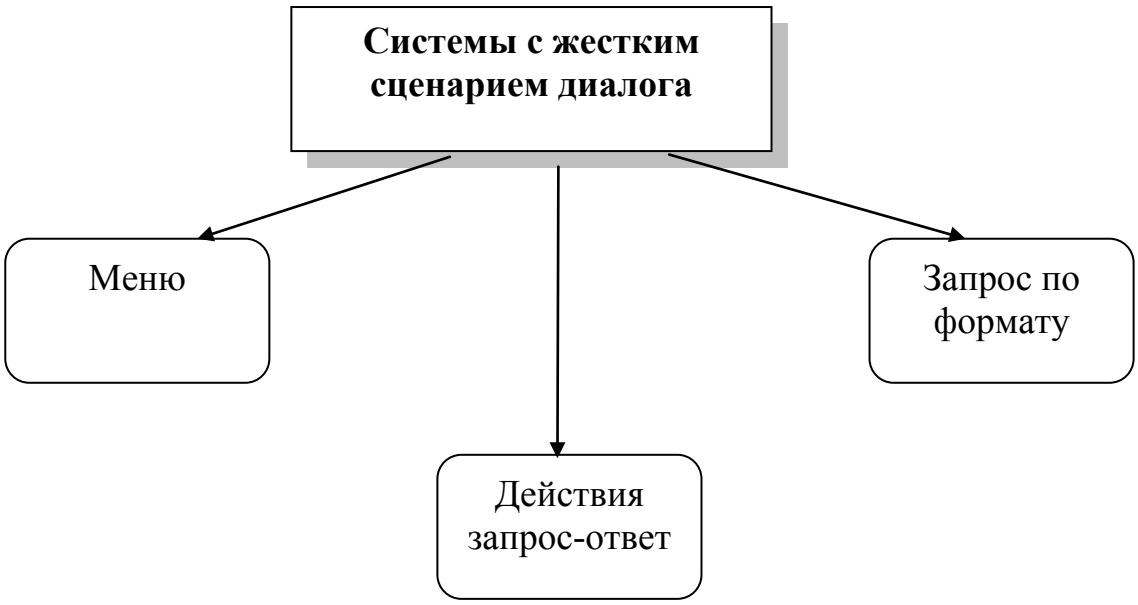
```
Program Main;
 Uses
   Sort;       {пользовательский модуль}
 Var
   X:mas; { mas - глобальный тип, описанный в модуле sort}
   n,i,j:integer;
   z:real;
 Begin
   Writeln('Введите размер массива');
   ReadLn (n); { размер массива }
   Writeln('Введите массив');
   For i := 1 to n do
      ReadLn (x[i]);
   SortY (x,n);
   Z := SumMas (x,n);
   Writeln('Сумма элементов массива = ',Z:8:2);
   Writeln('Упорядоченный массив');
   For i := 1 to n do
      Write (x[i]:8:2);
   Writeln;
   Writeln ('Конец работы. Нажмите клавишу ENTER');
   ReadLn;
 End.
```

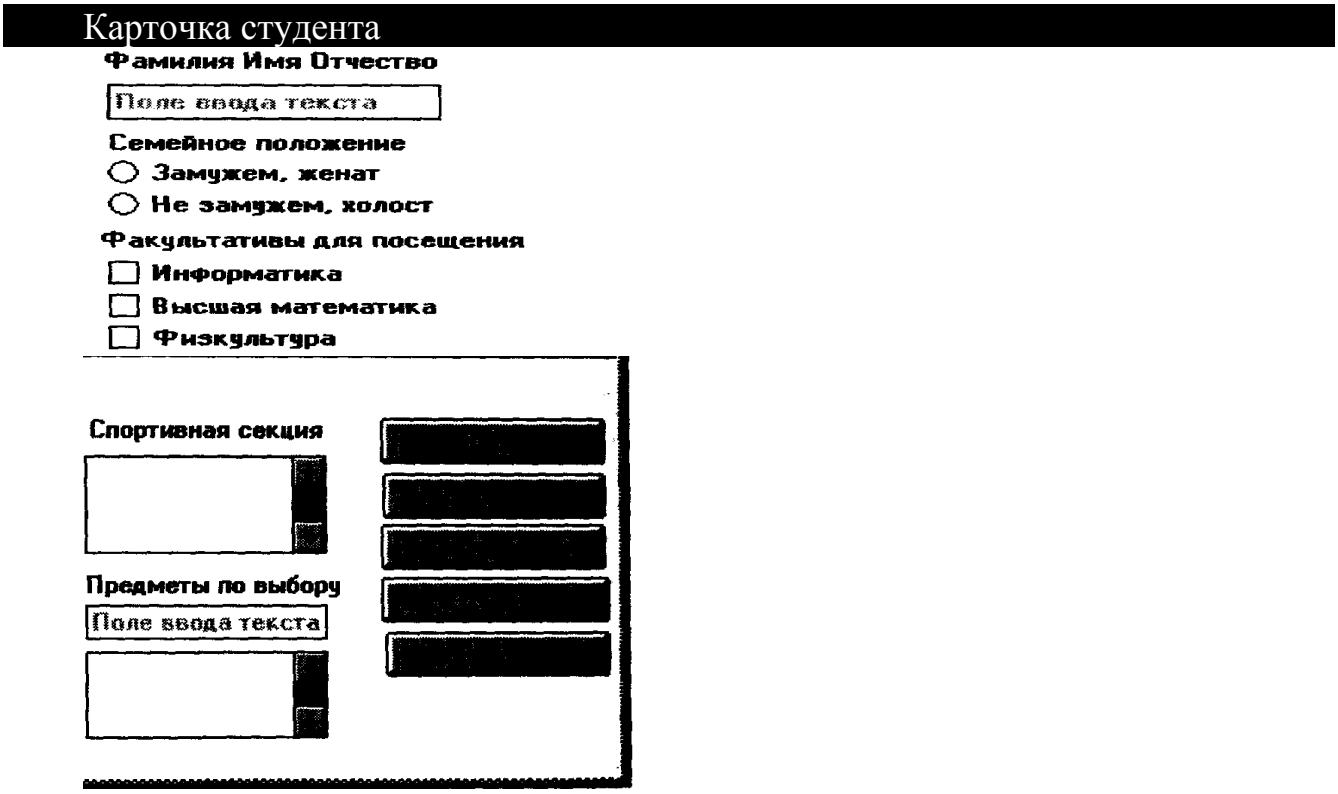
2. Организация пользовательского интерфейса.

Классификация диалоговых систем



Виды диалоговых систем с жестким сценарием диалога





3. Варианты заданий для выполнения лабораторной работы.

1. Разработать и отладить программу, состоящую из модуля, который содержит процедуру нахождения сумму положительных элементов массива вещественных чисел, и функцию, определяющую сумму элементов, находящихся после первого положительного элемента.

2. Разработать и отладить программу, состоящую из модуля, который содержит процедуру нахождения суммы отрицательных элементов массива и функцию, которая упорядочивает в порядке не возрастания элементы массива, стоящие до 2-го отрицательного.

3. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, если у матрицы больше половины строк начинается с отрицательного элемента, то все ее отрицательные элементы заменяет на 0 и функцию, которая компоненты массива, стоящие до первого отрицательного, увеличивает на 1.

4. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, если наибольший диагональный элемент матриц меньше 1 то все элементы матриц домноживает на 10 и функцию, которая компоненты массива, стоящие после второй отрицательной, уменьшает на 1.

5. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, обнуляет элементы матрицы под главной диагональю и функцию, которая определяет среднеарифметическое отрицательных компонент, стоящих до максимального элемента в массиве.

6. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, меняет местами первый и последний столбцы матрицы и функцию, которая, первый отрицательный элемент массива меняет местами с максимальным элементом массива.

7. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, находит суммы элементов тех строк матриц, у которых на главной диагонали положительный элемент и функцию, определяющую сумму элементов, находящихся после первого отрицательного элемента.

8. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, подсчитывает сумму элементов по каждой строке, следующих за первым положительным элементом и функцию, определяющую среднеарифметическое отрицательных компонент, стоящих до максимального элемента в массиве.

9. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, находит среднеарифметическое отрицательных компонент, расположенных после максимального элемента в массиве и функцию, определяющую среднеарифметическое положительных компонент, стоящих до максимального элемента в массиве.

10. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, определяет количество компонент массива, стоящих между максимальной и минимальной компонентой и функцию, определяющую сумму положительных компонент, стоящих до минимального элемента в массиве.

11. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, определяет сумму компонент массива, стоящих между максимальной и минимальной компонентой и функцию, определяющую количество положительных компонент, стоящих до максимального элемента в массиве.

12. Разработать и отладить программу, состоящую из модуля, который содержит процедуру, которая, все компоненты векторов с нечетными номерами меняет их квадратами, а с четными номерами - множит на 10 и функцию, которая, первый

отрицательный элемент массива меняет местами с максимальным элементом массива.

13. Напишите модуль, содержащий процедуру нахождения максимального элемента массива вещественных чисел, и функцию, определяющую минимальный элемент.

14. Разработать и отладить программу, состоящую из модуля, который содержит процедуру нахождения среднего арифметического положительных элементов массива вещественных чисел, и функцию, определяющую сумму его элементов, находящихся под главной диагональю.

15. Разработать и отладить программу, состоящую из модуля, который содержит процедуру нахождения среднего арифметического отрицательных элементов массива вещественных чисел, и функцию, определяющую сумму его элементов, находящихся над главной диагональю.

16. Дана матрица $[A_{i,j}]_{n,m}$. Определить количество элементов в каждой строке, находящееся между первыми двумя отрицательными элементами строки. Из полученных значений сформировать вектор.

17. В строке матрицы, в которой находится минимальный элемент матрицы, произвести следующее преобразование: вычесть из всех элементов этой строки максимальный элемент этой матрицы и подсчитать сумму результатной матрицы.

18. Дана матрица, $[A_{i,j}]_{n,m}$. Найти разность между минимальным и максимальным значением отрицательных элементов матрицы.

19. Дана матрица$[A_{i,j}]_{n,m}$. Подсчитать сумму минимальных элементов столбцов, в которых есть хотя бы один отрицательный элемент.

20. Даны два массива A и B. Определить вектора $[b_i]_n$ и $[c_i]_n$ компонентами которых соответственно являются среднеарифметические отрицательных элементов строк.

21. Даны два массива A и B. Найти суммы элементов тех строк матриц, у которых на главной диагонали положительный элемент. Указать номера этих строк.

22. Даны два массива A и B. Составить вектора компонентами которых являются суммы максимального и минимального элемента соответствующих строк матриц.

23. Даны два массива A и B. Подсчитать сумму и количество элементов по каждой строке, следующих за первым положительным элементом. Из полученных значений сформировать вектора.

24. Даны два массива A и B. Подсчитать сумму и количество элементов по каждой строке, предшествующих первому отрицательному элементу. Из полученных значений сформировать вектора.

25. Даны два массива A и B. Переставить строки матрицы так, что бы элементы первого столбца были упорядочены в порядке возрастания значении элементов.

4. Контрольные вопросы.

1. Что собой представляет архитектурный подход к разработке программы?
2. Назовите методы контроля структуры программы.
3. Какие требования предъявляют пользователи к интерфейсу пользователя?
4. Какие диалоговые системы и графический интерфейс используются для разработки Ваших программных продуктов?

Использованная литература:

1. Абрамян М.Э. Programming Taskbook. Ростов-на-Дону, 2005.
2. Златопольский Д.М. Сборник задач по программированию. Санкт-Петербург, 2007.
3. ЛахтинА.С., Искакова Л.Ю. Языки и технология программирования. Екатеринбург, 1998.
4. Меньшиков Ф. Олимпиадные задачи по программированию. СПб.: Питер, 2006.
5. Шень А. Программирование. Теоремы и задачи. Москва, 2004.