МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Нижегородский государственный технический университет им. Р.Е. Алексеева»

Кафедра "Информационные радиосистемы"

Разработка алгоритмов для решения задач на ЭВМ

Методические указания к лабораторной работе
по дисциплине «Информационные технологии» для студентов
направления подготовки бакалавра 210400 «Радиотехника»
дневной формы обучения

Нижний Новгород 2012

Составитель Е.Н.Приблудова

УДК 621.325.5-181.4

Разработка алгоритмов для решения задач на ЭВМ: метод. указания к лаб. работе по дисциплине «Информационные технологии» для студентов направления подготовки бакалавра 210400 «Радиотехника» дневной формы обучения / НГТУ; Сост.: Е.Н.Приблудова. Н.Новгород, 2012, 16 с.

Изложены краткие сведения о формах представления алгоритма. Приведены примеры разработки алгоритмов. Сформулированы задания для выполнения лабораторной работы.

Редактор Э.А.Жирнова

Подп. к печ. . Формат 60х84 1/16. Бумага газетная. Печать офсетная. Печ.л. 1. Уч.-изд.л. 0,7. Тираж экз. Заказ .

Нижегородский государственный технический университет им. Р.Е.Алексеева
Типография НГТУ. 603950, Н.Новгород, ул.Минина, 24.

ЦЕЛЬ РАБОТЫ

Научить студентов правильно разрабатывать алгоритмы, основными этапами которых являются:

- постановка задачи;
- построение математической модели;
- планирование структуры данных;
- проектирование программы;
- тестирование проекта;
- кодирование;
- проверка программы;
- составление документации.

КРАТКИЕ СВЕДЕНИЯ

1. Понятие алгоритма

Алгоритм **–** это точная последовательность действий, которые необходимо совершить для решения задачи, однозначно определяемая исходными данными.

Характеристики алгоритма:

- *определенность* подразумевает предсказуемость результата при заданных исходных данных;
- *дискретность* – возможность разделения алгоритма на отдельные элементарные действия;
- *результативность* означает, что через некоторое конечное число шагов алгоритм выдаст результат (цель достигнута, задача решена; цель недостижима, задача не имеет решения);
- *массовость* предполагает, что алгоритм даст результат для любых исходных данных из некоторого их множества.

2. Формы представления алгоритма

Первая форма - ***словесное описание*** алгоритма решения задачи.
Вторая форма – представление алгоритма ***совокупностью математических формул***.
Третья форма - запись алгоритма с использованием систем обозначений и правил, называемых ***псевдокодом*** (язык проектирования программ).

Четвертая форма - графическая форма записи алгоритма, называемая ***блок-схемой***.

2.1. Декларативная и исполняемая части при записи алгоритма с помощью псевдокода

Алгоритм, представленный псевдокодом, должен иметь ***декларативную*** и ***исполняемую*** части.

Декларативная часть состоит из заголовка алгоритма и объявлений. В заголовке за ключевым словом (ключевые слова выделяются полужирным шрифтом и используются в сокращенной форме) **алгоритм** (или **алг**) нужно указать имя алгоритма. Имя выбирается так, чтобы было ясно, какая задача решается. Далее с новой строки могут следовать объявления. Чтобы указать тип данного, перед его именем пишут сокращенное ключевое слово: **компл** для "комплексный", **вещ** для "вещественный", **двточн** для "двойной точности", **цел** для "целый", **лог** для "логический", **симв** для "символьный", **строк** для "строковый" и т. д.

Объявление типа переменных можно записать так:

тип $v_1, v_2, \ldots, v_n$

где $v_1, v_2, \ldots, v_n$ - имена переменных.

Здесь же в декларативной части некоторым или всем переменным можно задать начальные значения:

тип $v_1 \leftarrow a_1, v_2 \leftarrow a_1, \ldots, v_n \leftarrow a_1$

В этом объявлении $a_1, a_2, \ldots, a_n$ - константы; знак $\leftarrow$ читается "присвоить значение". Например, фраза $v_1 \leftarrow a_1$ означает переменной v_1 назначить значение a_1.

Исполняемая часть алгоритма начинается за словом **начало**. Она состоит из предложений, каждое из которых записывается с новой строки. Роль предложений играют команды, т. е. предписания выполнить отдельное законченное действие. Завершается алгоритм словом **конец.**

Алгоритм может иметь комментарии. Текст каждого из них должен размещаться между символами /* и */. Итак, общий вид алгоритма, записанного псевдокодом, таков:

алгоритм *имя*
  объявления
начало
  команды
конец

Исходные данные можно задать командами присваивания, объявления типа или командами ввода.

Команда ввода

ввести $v_1, v_2, \ldots, v_n$

присваивает переменным, элементам массива значения исходных данных, набираемых на клавиатуре.

Команда вывода

вывести $a_1, a_2, \ldots, a_n$

передает на внешний носитель информации (экран, принтер) значения выражений $a_1, a_2, \ldots, a_n$ числового, символьного, строкового и логического типов.

2.2. Базовые элементы блок-схемы для составления алгоритма

На рис. 1 приведены базовые элементы блок-схемы, из которых выстраивается алгоритм решения задачи.

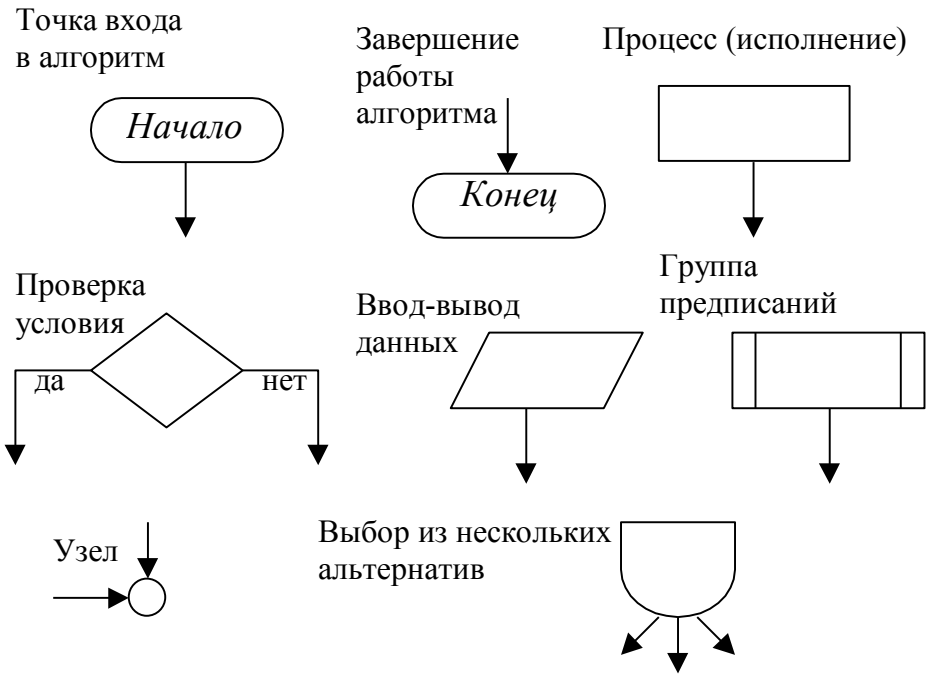

Рис. 1. Базовые элементы блок-схемы

3. Основные управляющие структуры представления алгоритма псевдокодом и блок-схемой

3.1. Линейный алгоритм

В алгоритмическом языке ***линейным*** является ***алгоритм***, состоящий из последовательности простых команд (ввода / вывода и присваивания). Команды в записи алгоритма располагаются в том порядке, в каком должны быть выполнены предписываемые ими действия.

Последовательность команд образует составную команду "цепочка", которая в записи псевдокодом и блок-схемой имеет вид, приведенный на рис. 2.

команда$_1$
команда$_2$
…
команда$_n$

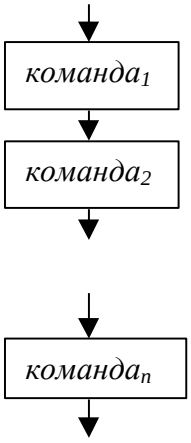

Рис. 2. Команда "цепочка" в записи псевдокодом и блок-схемой

3.2. Разветвляющийся алгоритм

Любое свойство величин, которое может соблюдаться или не соблюдаться для их текущих значений (число может быть четным или нечетным, одно число может быть меньше или не меньше другого, быть кратным или некратным некоторому значению и т. п.), называется **условием.**

Если в алгоритме присутствует операция проверки условия, то такой алгоритм – ***разветвляющийся алгоритм***.

В алгоритмическом языке отношения представляют равенства и неравенства математики. В них используются общепринятые знаки: $>, <, \geq, \leq, =, \neq$.

Простые условия можно соединять союзами **и, или**. Условия такого вида называются ***составными***. Пусть b_1 и b_2 - условия. Составное условие b_1 **и** b_2 истинно, если соблюдается вместе и b_1, и b_2 (например, двойное неравенство $1 \leq x \leq 2$, записываемое в алгоритмическом языке условием $1 \leq x$ и $x \leq 2$, будет истинным, если $x \in [1;2]$, и ложным, если $x \notin [1;2]$). Условие b_1 **или** b_2 истинно, когда соблюдается, по крайней мере, одно из условий b_1 или b_2, не-

важно какое. Перед условием можно написать частицу не. Тогда, например, условие **не** b_1 истинно, если b_2 - ложно, и наоборот.

Программист, разрабатывая алгоритм, нередко сталкивается с необходимостью запрограммировать альтернативу и, таким образом, выполнить одно или другое действие в зависимости от сформулированного условия. Эту проблему помогает решить команда ***ветвление***. Она имеет две формы – ***полную*** и ***сокращенную***.

Полная команда ветвления в записи псевдокодом и блок-схемой имеет вид (рис. 3):

если *условие*
то
  серия$_1$
иначе
  серия$_2$
конец-если

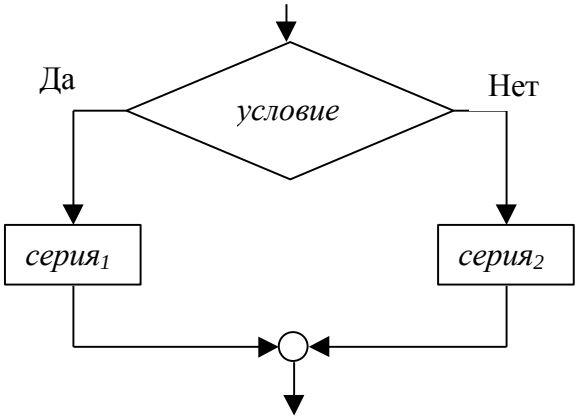

Рис. 3. Полная команда ветвления в записи псевдокодом и блок-схемой

Полная команда ветвления работает следующим образом. Сначала проверяется, соблюдается ли *условие*. Если оно соблюдается, то выполняется *серия*$_1$, и на этом работа команды заканчивается. Если же *условие* не соблюдается, то выполняется *серия*$_2$, после чего работа команды завершается. Команды *серий* реализуются подряд, каждая по своим правилам.

Сокращенная команда ветвления в записи псевдокодом и блок-схемой имеет вид (рис. 4):

если *условие*
то
  серия
конец-если

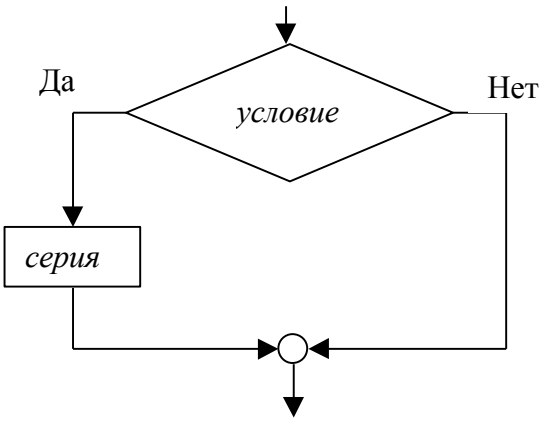

Рис. 4. Сокращенная команда ветвления в записи псевдокодом и блок-схемой

Данная команда предполагает, что при соблюдении *условия* следует выполнить *серию*, а при несоблюдении – пропустить *серию* и перейти к выполнению команды, следующей за ключевым словом **конец-если.**

Многозначное ветвление

Нередко приходится выбирать не из двух, а из нескольких возможностей. Такую ситуацию (многозначное ветвление) можно описать вложенными друг в друга командами ветвления или *командой выбора*. Она в записи псевдокодом и блок-схемой (рис. 5) имеет вид:

выбор *выражение*
  список значений$_1$ **выполнять** *серия*$_1$
  список значений$_2$ **выполнять** *серия*$_2$
  . . .
  список значений$_k$ **выполнять** *серия*$_k$
  иначе **выполнять** *серия*$_0$
конец-выбор

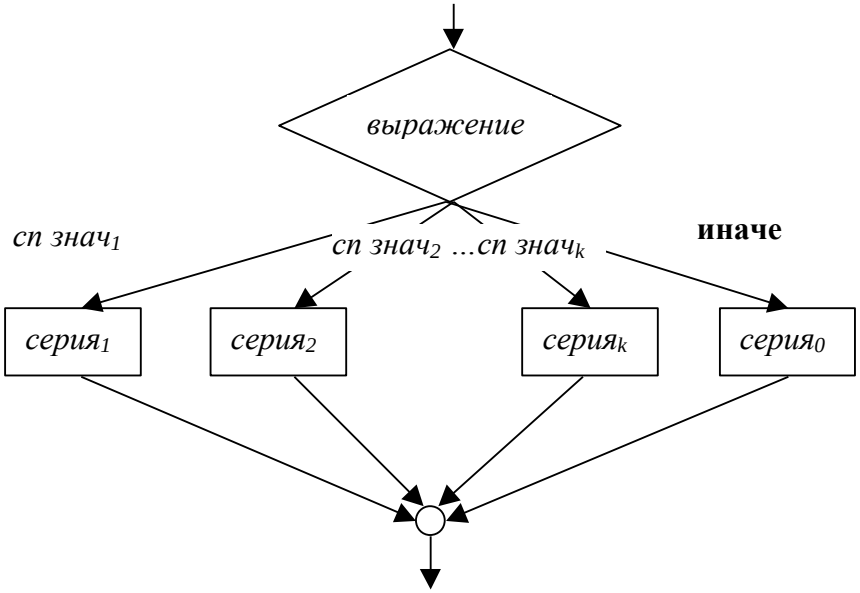

Рис. 5. Команда выбора в записи псевдокодом и блок-схемой

Команда выбора работает так. Сначала определяется значение *выражения*. Если оно принадлежит *списку значений*$_i$ *(i=1, 2, …, k)*, то выполняются команды *серии*$_i$, и на этом работа всей команды завершается. В противном случае (значение не принадлежит ни одному *списку значений*$_i$) работают команды *серии*$_0$, указанной после слова **иначе.**

Пример. Рассмотрим задачу решения квадратного уравнения. В качестве исходных данных примем значения параметров a, b, c. Требуется вычислить действительные корни уравнения $ax^2 + bx + c = 0$ при $a \neq 0$.

Приведем текст алгоритма в записи псевдокодом:

алгоритм Корни уравнения
  вещ $a, b, c, d, ds, x1, x2$
начало
  ввести a, b, c
  если $a \neq 0$
  то
    $d \leftarrow b^2 - 4ac$
      если $d >= 0$
      то
        $ds \leftarrow \sqrt{d}$
        $x1 \leftarrow (-b + ds)/(2a)$
        $x2 \leftarrow (-b - ds)/(2a)$
        вывести значения $x1, x2$
      иначе
        вывести сообщение: Решения нет
      конец-если
  иначе
    вывести сообщение об ошибке
  конец-если
конец

Приведем блок-схему алгоритма решения данной задачи:

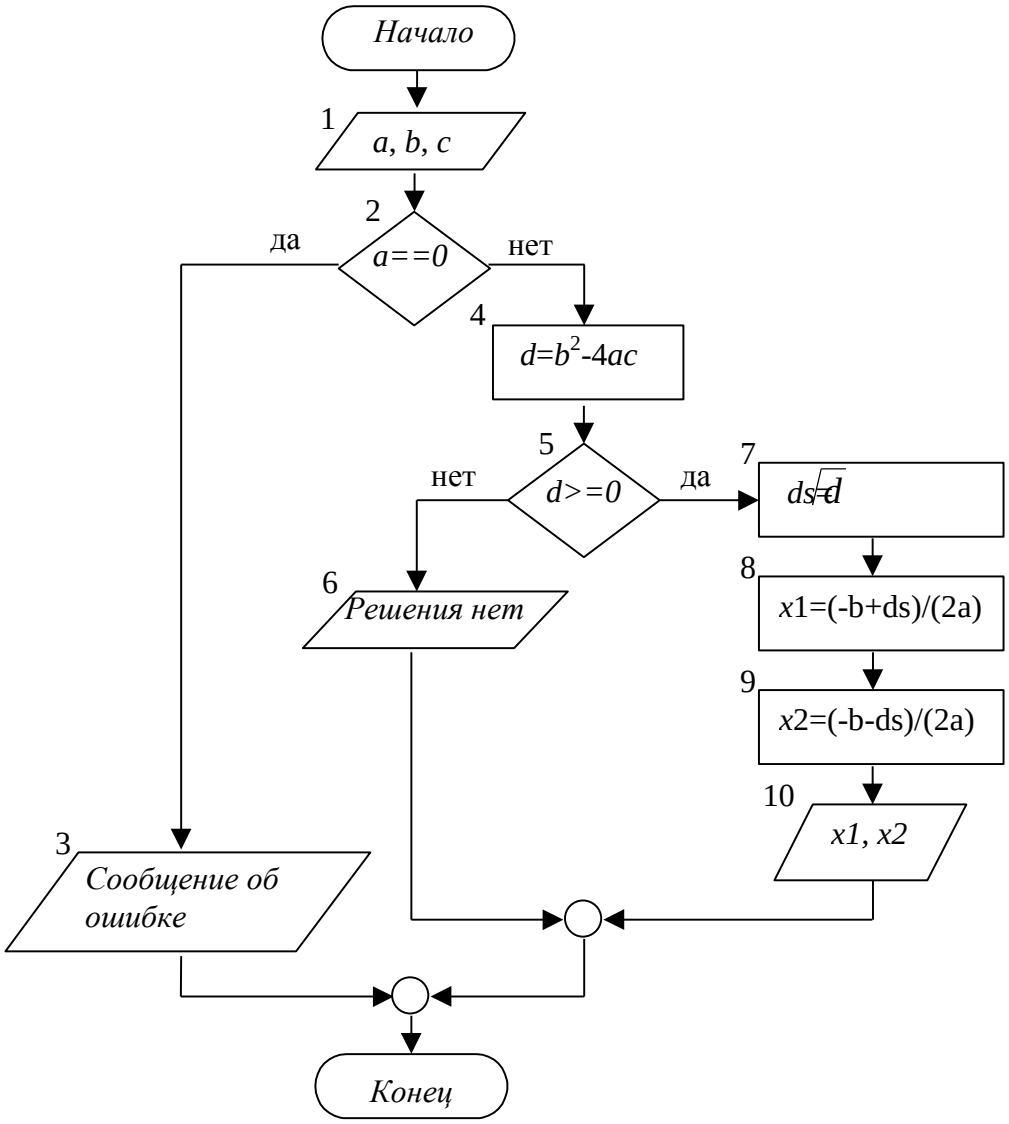

Рис. 6. Блок-схема алгоритма решения задачи

Номер блока	Описание
1	Ввод исходных данных
2	Проверка значения параметра *a*
3	Вывод сообщения об ошибке
4	Вычисление дискриминанта
5	Проверка наличия действительных корней
6	Вывод сообщения об отсутствии корней
7	Вычисление квадратного корня из дискриминанта
8	Вычисление первого корня
9	Вычисление второго корня
10	Вывод найденных значений корней уравнения

Составленный алгоритм можно использовать неоднократно с различными значениями исходных данных. Всегда необходимо стремиться к тому, чтобы составленный алгоритм не был привязан к конкретным значениям каких-либо параметров. Например, мы могли в принципе изначально считать, что значения параметров a, b, c таковы, что удовлетворяют условию $d \geq 0$. Однако, это предположение существенно сужает множество задач, которые можно будет решать по этому алгоритму.

3.3. Циклический алгоритм

Группа последовательных действий, завершающаяся переходом при выполнении некоторого условия на первую команду этой группы, называется ***циклом***.

Если алгоритм содержит предписание многократного исполнения одной и той же последовательности операций, то такой алгоритм называют ***циклическим***.

Существуют циклы с параметром или без него, с постоянным или переменным начальным, конечным значениями и шагом изменения.

Рассмотрим структуру ***цикла с предусловием***. Данный цикл можно описать командами **пока выполнять** и **для выполнять**. Это составные команды. Первая из них имеет вид, записанный псевдокодом и представленный графически на рис. 7.

пока *условие* **выполнять**
  серия
конец-цикл

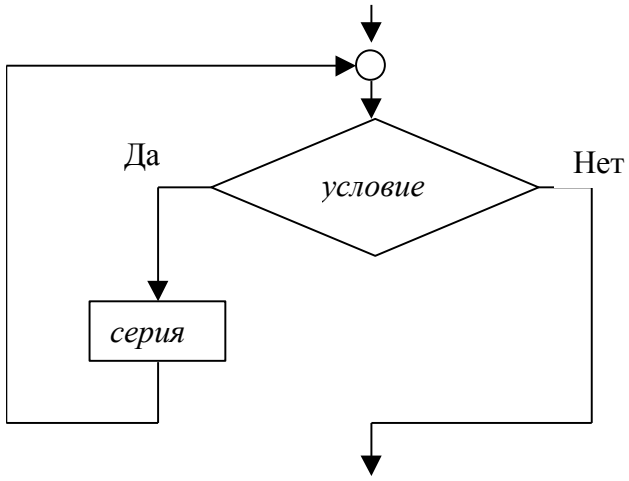

Рис. 7. Цикл с предусловием в записи псевдокодом и блок-схемой

Приведем правила ее работы: *серия* выполняется, пока *условие* истинно; если же с самого начала *условие* не соблюдается, *серия* не выполняется ни разу.

Вид простой команды **для выполнять** в записи псевдокодом представлен ниже:

для i **от** $j_н$ **до** $j_к$ [**шаг** h] **выполнять**
  серия
конец-цикл

Здесь квадратные скобки показывают, что элемент, заключенный в них, может отсутствовать.

Цикл с постусловием можно описать командой **выполнять пока.**
В записи псевдокодом и блок-схемой (рис. 8) она имеет вид:

выполнять
  серия
пока *условие*

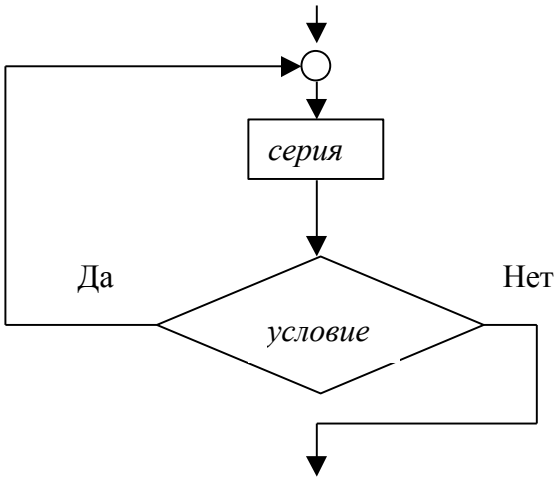

Рис. 8. Цикл с постусловием в записи псевдокодом и блок-схемой

Здесь *серия* выполняется один раз, каким бы ни было начальное значение *условия*, а затем повторяется до тех пор, пока *условие* остается верным.

Пример. Решим задачу табулирования (составление таблиц значений функции): для значений аргумента x построим таблицу значений полинома

Чебышева третьего порядка $T(x) = 4x^3 - 3x$. Пусть $a = -1, b = 1, n = 11$, где $[a,b]$ - интервал изменения аргумента x, n - количество точек разбиения отрезка.

Разработаем алгоритм. Зададим его аргументы (**ввести** a, b, n). Табулирование функций является циклическим алгоритмом, в котором количество повторений цикла (n - число строк таблицы) известно. Подсчитаем шаг h изменения x по формуле

$h = (b - a)/(n - 1)$ при $n > 1$.

Управлять циклом будет переменная i, которая принимает начальное значение, равное единице, и изменяется до n включительно с шагом, равным единице. Текущее значение x будем находить по формуле

$x = a + (i - 1)h$, где $i = 1, 2, \ldots, n$.

Текст алгоритма в записи псевдокодом:

алгоритм Табулирование
  вещ a, b, h, x, T
  цел i, n
начало
  ввести a, b, n
  вывести заголовок задачи, a, b, n и "шапку" таблицы
  $h \leftarrow (b - a)/(n - 1)$
  для i **от** 1 **до** n **шаг** 1 **выполнять**
    $x \leftarrow a + (i - 1)h$
    $T \leftarrow (4x^2 - 3)x$
    вывести x, T
  конец-цикл
  вывести основание таблицы
конец

Спроектируем структуру выводных данных. Вывод значения x выполним в формате $N.N$, а значения $T(x)$ - в формате $N.NNN$, где символ N заменяет одну из цифр $0, 1, \ldots, 9$. Строкой из минусов отделим исходные данные от выходных, а строками из знаков = (вверху и внизу) укажем границы области вывода.

Тогда получится таблица следующего вида:

= =

Таблица функции $T(x) = 4x^3 - 3x$.

Исходные данные:

$a = -1.0$ $b = 1.0$ $n = 11$

- -

Ответы:

Полином Чебышева

x	$T(x)$
$N.N$	$N.NNN$

= =

4. Понятие массива

Массив – это набор пронумерованных однотипных данных.

При объявлении массива задается тип его элементов (**компл**, **вещ**, **двточн**, **цел**, **лог**, **симв**, **строк** и т. п.), затем ключевое слово **массив**, потом имя массива, за которым в квадратных скобках стоит размер массива.

Пример. Решим следующую задачу: на плоскости на расстояниях $\rho_1, \rho_2, \ldots, \rho_n$ от центра кругового кольца с радиусами r_1 и r_2 расположены точки. Необходимо определить количество точек, попавших внутрь кольца.

Решение начнем с построения математической модели. Из расстояний $\rho_1, \rho_2, \ldots, \rho_n$ образуем массив $P = (\rho_i)$ $(i = 1, 2, \ldots, n)$. Условия попадания i-й точки внутрь кольца запишем в виде двойного равенства $r_1 < \rho_i < r_2$ $(i = 1, 2, \ldots, n)$. В алгоритмической записи оно имеет вид $r_1 < \rho_i$ **и** $\rho_i < r_2$. Пусть число точек на плоскости равно десяти $(n = 10)$.

Теперь разработаем алгоритм. Введем ограничение на максимально возможное число элементов массива. Объявим, **вещ массив** $P[1000]$ заведомо большей длины, чем фактический массив $P[n]$ длины n.

Вот шаги алгоритма: объявим переменные и массив, подготовим исходные данные и приступим к проектированию цикла. Обозначим через k счетчик числа точек, находящихся внутри кольца. Его начальное значение должно быть равно нулю, поэтому в подготовке цикла предусмотрим действие $k \leftarrow 0$. Обозначим через i перебор точек. Для каждой точки будем проверять условие $r_1 < \rho_i$ **и** $\rho_i < r_2$. Если на очередной итерации оно соблюдается, увеличим счетчик k на единицу. Перебор значений ρ_i выполняется, пока истинно неравенст-

во $i \le n$. Когда i превзойдет значение n, выйдем из цикла, т. е. перейдем к следующему по порядку действию – выводу результата работы алгоритма.

алгоритм *Мишень*
  цел n, k, i
  вещ массив $P[1000]$
  вещ r_1, r_2
начало
  ввести n, массив P и радиусы r_1, r_2 /* Исходные данные */
  вывести заголовок задачи и значения n, P, r_1, r_2
  $k \leftarrow 0$
  для i **от** 1 **до** n **шаг** 1 **выполнять** /* Подсчитать число точек */
    если $r_1 < \rho_i$ **и** $\rho_i < r_2$
    то
      $k \leftarrow k + 1$
    конец-если
  конец-цикл
  вывести заголовки выходных данных и значений k
конец

Теперь спланируем тестирование. Проверку программы выполним при следующих исходных данных.

Зададим число точек на плоскости:

$n = 10$.

Подберем расстояния и радиусы:

$P =$ 12.34 56.78 90.98 7.6 0.54

0.03 21.0 12.89 7.41 99.99

$r_1 = 12.35$ $r_2 = 56.90$.

Выходные данные в этой задаче будут иметь вид:

Мишень

Исходные данные:

$n =$ 10

Массив $P[n]$:

12.34 56.78 90.98 7.6 0.54

0.03 21.0 12.89 7.41 99.99

$r_1 = 12.35$ $r_2 = 56.90$

Ответ:

внутри 3 точки (три точки находятся внутри кольца).

ЗАДАНИЯ И ПОРЯДОК ВЫПОЛНЕНИЯ

Варианты для заданий № 1, 2 определяются преподавателем.

Алгоритмы необходимо оформить в текстовом редакторе MS Word в виде объекта **Рисунок MS Word** с помощью команды **Вставка** -> **Объект** -> **Рисунок MS Word**.

Задание № 1

Составить разветвляющийся алгоритм для решения задачи с использованием следующих форм представления алгоритма: ***псевдокод*** и ***блок-схема***.

Задание № 2

Составить циклический алгоритм для решения задачи с использованием одной формы представления алгоритма: ***псевдокод***.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое алгоритм?
2. Перечислите характеристики алгоритма.
3. Какие формы представления алгоритма Вам известны?
4. Перечислите основные ключевые слова при записи алгоритма с помощью псевдокода.
5. Какие базовые элементы для составления алгоритма в виде блок-схемы Вам известны?
6. Что представляет собой линейный алгоритм?
7. Что представляет собой разветвляющийся алгоритм?
8. Запишите полную и сокращенную команды ветвления.
9. Что означает многозначное ветвление?
10. Что такое цикл?
11. Какими командами описываются циклы с предусловием и с постусловием?
12. Что такое массив?

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Информатика. Базовый курс / Симонович С.В. и др. – СПб.: Питер, 2007.
2. Кнут Д.Э. Искусство программирования: Пер.с англ.:В 3-х т. Т.1: Основные алгоритмы / Д. Э. Кнут. - 2-е изд. - М.: Вильямс, 2003.