Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

**Санкт-Петербургский государственный технологический
университет растительных полимеров**

П.Е.Антонюк

АЛГОРИТМИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

**Учебно-методическое пособие
для студентов I - II курсов**

Санкт-Петербург
2009

П.Е.Антонюк

АЛГОРИТМИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Учебно-методическое пособие

для студентов I - II курсов

Санкт-Петербург
2009

Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ РАСТИТЕЛЬНЫХ
ПОЛИМЕРОВ

П.Е.Антонюк

АЛГОРИТМИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

Учебно-методическое пособие
для студентов I-II курсов

Санкт-Петербург
2009

ББК 73я7
А456
УДК 681.3(075)
Алгоритмические основы информатики: учебно–методическое пособие для студентов I – II курсов/ сост. П.Е.Антонюк; ГОУВПО СПбГТУРП. – СПб., 2009 – 72 с.

Настоящее пособие подготовлено в соответствии с программой дисциплины "Информатика". Содержит теоретические сведения и описания лабораторных работ, выполняемых студентами I и II курсов.

Предназначено для студентов очного, вечернего и заочного обучения специальностей 140604 и 220301.

Рецензенты: д-р техн. наук, зав. кафедрой информационных технологий и систем управления СПбГТУРП Г.А. Кондрашкова;

канд. техн. наук, доцент кафедры информатики Санкт-Петербургского государственного университета сервиса и экономики О.Н. Казанцева.

Подготовлено и рекомендовано к печати кафедрой прикладной математики и информатики ГОУВПО СПбГТУРП (протокол № 5 от 19.12.2008).

Утверждено к изданию методической комиссией факультета АСУТП ГОУВПО СПбГТУРП (протокол № 8 от 22.12.2008).

ВВЕДЕНИЕ

Настоящее пособие предназначено для изучения базовых алгоритмических структур информатики. Для каждой структуры приводится подробное описание – как графическое, так и на языке программирования. Выбор в качестве языка программирования языка Паскаль объясняется его широким распространением, а также, по мнению автора, большими возможностями для последующего перехода на объектно-ориентированный язык программирования Delphi.

После подробного объяснения структур приведено описание четырех лабораторных работ. В каждой из работ рассматривается выполнение контрольного примера, по образцу которого студентам необходимо будет сделать свой вариант работы.

Выполнение каждой лабораторной работы должно заканчиваться предоставлением отчёта, который должен содержать структурную схему задачи, исходные данные и результат выполнения задачи. Пример оформления отчета приведен в Приложении 1.

1. ПОНЯТИЕ АЛГОРИТМА

Понятие алгоритма такое же основополагающее для информатики, как и понятие информации. Именно поэтому важно в нем разобраться.

Алгоритм – это система формальных правил, однозначно приводящая к решению данной задачи. Если это определение сузить до области применения вычислительной техники, то можно определить алгоритм как последовательность арифметических и логических действий над числовыми значениями переменных, приводящую к вычислению решения задачи при изменениях исходных данных в достаточно широких пределах.

Название "алгоритм" произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда ибн Муса ал–Хорезми (Algorizmi), жившего в 783—850 гг. В своей книге "Об индийском счете" он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними "столбиком", знакомые теперь каждому школьнику. В XII в. эта книга была переведена на латынь и получила широкое распространение в Европе. Имя автора в латинизированной форме стало обозначать в средневековой Европе всю систему десятичной арифметики, а позже – и правила выполнения определенных действий, то есть алгоритмы.

Исполнитель алгоритма – это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом. Исполнителя характеризуют:

- среда;
- элементарные действия;
- система команд;
- отказы.

Среда (или обстановка) — это "место обитания" исполнителя. Например, для исполнителя Робота из школьного учебника среда — это бесконечное клеточное поле. Стены и закрашенные клетки тоже часть среды. А их расположение и положение самого Робота задают конкретное состояние среды.

Каждый исполнитель может выполнять команды только из некоторого строго заданного списка — системы команд исполнителя. Для каждой команды должны быть заданы условия

применимости (в каких состояниях среды может быть выполнена команда) и описаны результаты выполнения команды. После вызова команды исполнитель совершает соответствующее элементарное действие.

Отказы исполнителя возникают, если команда вызывается при недопустимом для нее состоянии среды. Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные команды, не задавая вопросов "почему" и "зачем". В информатике универсальным исполнителем алгоритмов является компьютер.

Как и любое научное понятие, алгоритм обладает некоторыми свойствами, характеризующими его выполнение.

Свойства алгоритма:

1) Детерминированность (определенность).

Каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче. Поэтому применение алгоритма к одним и тем же исходным данным должно приводить к одному и тому же результату.

2) Результативность (конечность).

За конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.

3) Массовость.

Алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

4) Дискретность (прерывность, раздельность).

Алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) этапов (шагов).

5) Понятность.

Исполнитель алгоритма должен понимать, как его выполнять. Другими словами, при наличии алгоритма и произвольного варианта исходных данных исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

Для составления алгоритма и программы решения задач на компьютере математическая формулировка задачи, включающая символы математического анализа (символы интеграла, производной, дифференциальных операторов и т.д.), должна быть преобразована непосредственно в процедуру решения задачи, представляющую собой последовательность арифметических действий и логических связей между ними, то есть должен быть выбран метод решения задачи. Например, при решении квадратного уравнения $ax^2 + bx + c = 0$ вычисление корней производится по формуле

$$x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})/(2a).$$

Формулы, подобные данной, могут составить основу алгоритмов решения подобных задач.

Недостаток формул заключается в том, что они не обладают достаточной массовостью. Для алгебраических уравнений степени $n \geq 5$ не существует общих формул, выражающих связь между коэффициентами уравнений и их корнями в радикалах. Поэтому в вычислительной математике разработаны методы арифметизации задач, называемые численными методами, которые приспособлены для решения широких классов задач и обеспечивают их достаточно высокую точность решения. Выбор того или иного численного метода для решения задачи на компьютере связан, с одной стороны, с требованиями, предъявляемыми постановкой задачи (точность решения, быстрота получения результата, стоимость подготовки программы решения задачи), и, с другой стороны, с требованиями, предъявляемыми компьютером и программой к численному методу для его компьютерной реализации.

Алгоритм, на основе которого составляется программа решения задачи, представляет собой последовательность действий, которые может выполнить (или может выполнить наиболее рационально) компьютер. В нем отражены не только арифметические действия, необходимые для реализации выбранного численного метода, но и логические связи, которые численный метод налагает на исходные данные. Эти логические

связи должны быть заданы в форме, воспринимаемой вычислительными машинами, – проверкой тех или иных соотношений, допускающих (или не допускающих) автоматическое выполнение действий, предписываемых машине программой.

2. СПОСОБЫ ОПИСАНИЯ АЛГОРИТМА

Алгоритм может быть представлен различными способами. На практике наиболее распространены следующие формы представления алгоритма:

– словесная (запись на естественном языке);
– графическая (изображения из графических символов);
– псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
– программная (тексты на языках программирования).

Словесный способ записи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Пример. Записать алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел (алгоритм Эвклида).

Алгоритм может быть следующим:

1) задать два числа;
2) если числа равны, то взять любое из них в качестве ответа и
3) остановиться, иначе продолжить выполнение алгоритма;
4) определить большее из чисел;
5) заменить большее из чисел разностью большего и меньшего из чисел;
6) повторить алгоритм с шага 2.

Описанный алгоритм применим к любым натуральным числам и должен приводить к решению поставленной задачи.

Словесный способ не имеет широкого распространения, так как такие описания:

- строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов. Псевдокод занимает промежуточное место между естественным и формальным языками. С одной стороны, он близок к обычному естественному языку, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой стороны, в псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи.

В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя.

Однако в псевдокоде обычно имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. В частности, в псевдокоде, так же, как и в формальных языках, есть служебные слова, смысл которых определен раз и навсегда. Они выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются.

Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным.

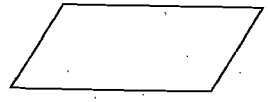
При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

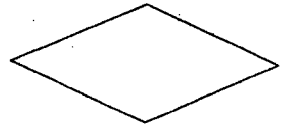
Все символы схем алгоритмов описываются государственным стандартом ГОСТ 19.701.90, в котором точно обозначены размеры каждого блока, его вид и назначение. В связи с учебным характером выполнения лабораторных работ в настоящем пособии, точное соблюдение размеров блоков схем алгоритмов необязательно (однако приветствуется).

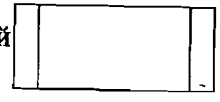
Далее приведены основные элементы структурных схем алгоритмов и их предназначение.

Пуск–останов:  : начало,конец алгоритма; вход/выход в подпрограмму.

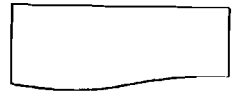
Процесс:  : вычислительное действие или последовательность действий.

Ввод–вывод:  : ввод-вывод в общем виде.

Решение:  : проверка условий.

Предопределенный процесс:  :вычисления по подпрограмме, стандартной подпрограмме.

Модификация:  :начало цикла.

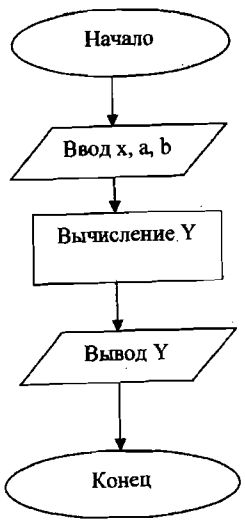
Документ:  :вывод результатов на печать.

3. ОСНОВНЫЕ ВИДЫ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Рассмотрим основные виды вычислительных процессов, встречающиеся в информатике.

1) *Линейный вычислительный процесс*: процесс, блоки которого выполняются последовательно один за другим (порядок выполнения блоков естественный).

Например, вычисление функции $Y = a * x + b$ описывается следующей структурной схемой



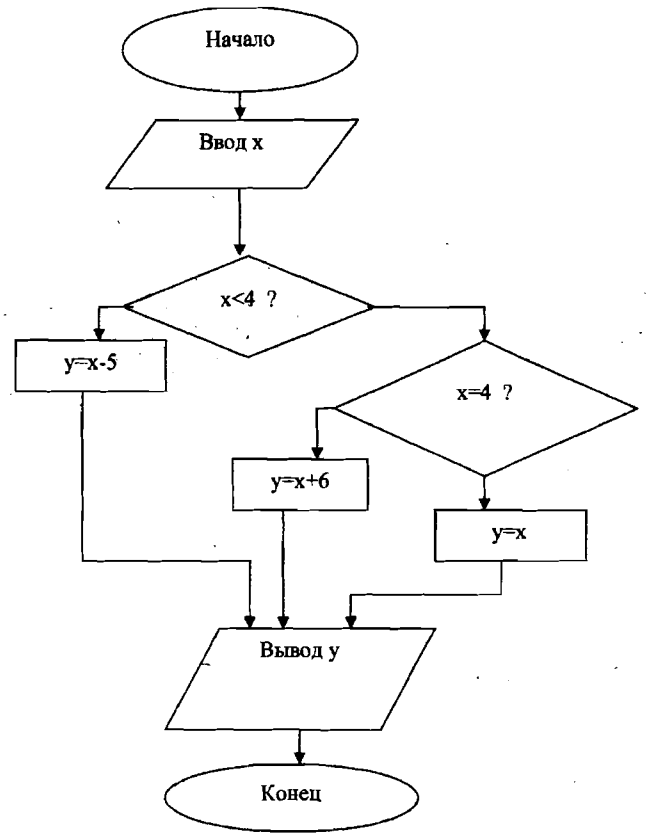
2)*Разветвляющийся вычислительный процесс*: процесс, который осуществляет вычисления в зависимости от выполнения того или иного логического условия по тем или иным формулам, по той или иной ветви.

В качестве примера рассмотрим вычисление функции:

$$y = \begin{cases} x-5, x<4 \\ x+6, x=4 \\ x, x>5 \end{cases}$$

В случае выполнения условия x<4, происходит вычисление функции y=x – 5, в противном случае происходит проверка условия x=4. Если данное условие выполняется, то вычисляется функция y

= x+6, иначе вычисляется функция y = x.

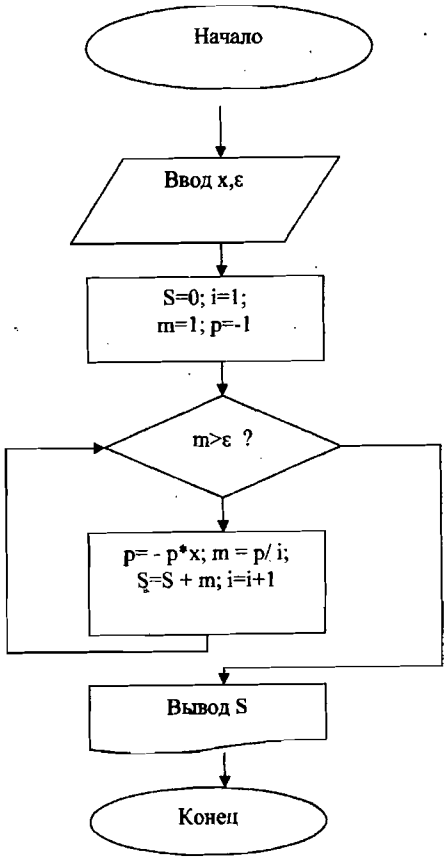


3) *Циклический вычислительный процесс*: такие процессы часто встречаются на практике в тех случаях, когда решение задачи сводится к многократному вычислению по одним и тем же математическим зависимостям при различных значениях входящих в них величин. Многократно повторяющиеся участки такого вычислительного процесса называют <u>циклами.</u> Циклический алгоритм позволяет существенно сократить объем программы за счет многократного выполнения ее циклического участка.

В качестве примера рассмотрим алгоритм вычисления

бесконечной суммы $S = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ...(-1)^{i-1}\frac{x^i}{i} + ...$ с заданной точностью ε (для данной знакочередующейся бесконечной суммы требуемая точность будет достигнута, когда очередное слагаемое станет по абсолютной величине меньше ε).

Схема выполнения алгоритма представлена ниже. На ней:
– p – числитель очередного слагаемого;
– m – очередное слагаемое;
– S – частичная сумма;
– i – номер очередного слагаемого.



4. ЛАБОРАТОРНЫЕ РАБОТЫ

Общая схема выполнения лабораторной работы

Выполнение каждой лабораторной работы включает несколько этапов, ни один из которых не должен пропускаться. Начинаться работа должна с разбора задания и составления списка объектов, которые будут встречаться в программе (в изображении алгоритма). Этот список объектов в виде таблицы идентификаторов (таблицы распределения памяти) очень важен в первых работах. Как показывает опыт составителя, неумение выделить и описать используемые в программе данные очень часто приводит к ошибкам в программах, например, порядковый номер путают со значением элемента в последовательности, общее количество объектов – с количеством объектов, удовлетворяющих какому-либо критерию, и тому подобное.

Изображение алгоритмов следует выполнять максимально подробно, особенно в первых работах. Нельзя пропускать изображения блоков алгоритма, если они "подразумеваются". Можно позволить объединять в блоке несколько однотипных действий (например, очистку нескольких переменных), но запрос на ввод какого-либо значения и, собственно, ввод этого значения в переменную с клавиатуры необходимо изображать двумя отдельными блоками, пока студенты не привыкнут, что всякому вводу данных пользователем программы должен предшествовать запрос со стороны программы на компьютере.

Каждая лабораторная работа должна быть выполнена на компьютере и только после этого может быть оформлена и представлена к защите. Отчет должен содержать:
– титульный лист,
– задание,
– таблицу идентификаторов,
– блок-схему алгоритма,
– листинг программы и протокол работы программы (в виде распечатки текстового файла с исходными данными и результатами работы программы).

Рекомендуемое оформление каждого раздела приведено при разборе контрольного варианта в каждой работе. Стоит отметить, что с первой же лабораторной работы следует придерживаться порядка и аккуратности в оформлении лабораторной работы. Это

важный фактор обучения алгоритмизации и составлению программ, так как такой стиль работы в дальнейшем существенно уменьшает количество ошибок и ускоряет процесс написания и отладки программ.

При наличии достаточного количества технических ресурсов (принтера и расходных материалов к нему) все части лабораторной работы (кроме блок-схемы алгоритма) могут распечатываться на принтере. Как минимум, в отчете должна присутствовать распечатка текста программы и результатов ее работы, остальные разделы могут быть выполнены вручную.

При написании текста программы следует придерживаться некоторых рекомендаций. В частности, текст программы должен содержать:

- Комментарий с указанием фамилии программиста и номера группы.
- Описание всех используемых в программе идентификаторов (имен).
- Ввод требуемого количества данных из указанного в задании файла или иной способ задания исходных данных, как сказано в задании.
- Вывод в выводной текстовый файл (до четвертой работы – на экран) введенных исходных числовых значений под заголовком "Исходные данные".
- Проведение обработки числового материала в соответствии с заданием.
- Вывод результатов обработки в выводной текстовый файл (или на экран) под заголовком "Результаты расчета", с пояснениями, если результат не может быть выведен.

Текст программы (листинг) необходимо писать с выделением вложенных циклов, составных операторов и структур смещением на две – три позиции, что облегчает чтение программы и упрощает отладку (поиск пропущенных программных скобок begin … end и несоответствий алгоритму).

Листинг и вывод результатов следует размещать так, чтобы в дальнейшем был возможен вывод на печать. Для этого следует учитывать, что ширина стандартного листа бумаги реально позволяет печатать текст шириной до 76 символов в строке (для листа формата А4).

В завершение, несколько слов о рекомендуемой литературе. Для работы желательно иметь описание языка и инструментальной среды используемой системы программирования. В частности, удобно пользоваться [1,2,3,4], где описано и то, и другое.

Лабораторная работа № 1.
Алгоритмы линейной структуры

Вопросы, изучаемые в работе

- Построение простейшей программы линейной структуры.
- Использование операторов присваивания и простейшего вывода для данных вещественного типа.
- Использование оператора описания переменных для данных вещественного типа.
- Изучение правил написания и вычисления арифметических выражений.
- Применение в выражениях встроенных математических функций языка Паскаль.

Задание (общее ко всем вариантам)

Запрограммировать вычисление заданной функции, вычислить и вывести на экран результат при указанных значениях аргументов. Проверить программу по приведенному в задании ответу. Оформить отчет по лабораторной работе в соответствии с образцом, приведенным для варианта № 26. Таблицу идентификаторов, блок-схему алгоритма и текст программы аккуратно выполнить от руки, с обозначениями пробелов (в программе), где они необходимы.

Требования к программе

- Программа должна содержать комментарий с указанием названия работы, номера варианта, фамилии студента и номера группы.
- Аргументы задавать операторами присваивания значений.
- При программировании выражений части выражений встречающиеся два и более раз, вычислять один раз с

запоминанием в промежуточных (рабочих) переменных.

- Вывод результата выполнять на экран по формату :12.
- Отладить программу, чтобы результат совпадал по всем цифрам с приведенным ответом.

Содержание программы

- Заголовок программы с комментарием.
- Описание переменных.
- Задание значений аргументам.
- Вычисление (если необходимо) промежуточных значений.
- Расчет результата.
- Вывод результата на экран.

Общие пояснения

Настоящая работа, как и все последующие, предполагает написание программы на языке Паскаль. Поскольку это первая работа по составлению программы, ниже приводится общая структура и правила написания программ на языке Паскаль.

Текст должен быть написан латинскими символами (прописные и строчные символы не различаются) в файле с именем, удовлетворяющим правилам файловой системы DOS и с расширением .PAS. Рекомендуется использовать имена, состоящие не более чем из 6 символов, за которыми следует номер лабораторной работы.

Операторы могут начинаться с любой позиции строки, переноситься на последующие строки (переход на другую строку разрешен в любом месте, где можно вставить пробел) и записываться по несколько операторов в одной строке. Операторы разделяются между собой символом ";".

Комментарии представляют собой любой текст (в том числе – кириллицей), заключенный в фигурные скобки. Он может занимать несколько строк и может вставляться внутрь любого оператора языка. Внутри комментария нельзя иметь фигурные скобки (вложенный комментарий).

Программа может начинаться с необязательного оператора PROGRAM <имя программы>, за которым должен идти блок описаний, состоящий из одного или нескольких разделов. Затем

идет выполняемый блок, заканчивающийся символом ".". Как правило, блок описаний содержит раздел описания переменных, начинающийся с ключевого слова VAR. Остальные разделы могут отсутствовать. Выполняемый блок должен быть заключен в операторные скобки BEGIN ... END, причем рекомендуется любую соответствующую пару скобок записывать, начиная с одной и той же колонки. Открывающую скобку Begin лучше всегда начинать с новой строки.

Пример простейшей программы будет приведен перед таблицей с данными вариантов заданий.

Уточним задачи лабораторной работы.

1. Программа линейной структуры – это программа, в которой каждое действие выполняется последовательно один и только один раз, т.е. в алгоритме присутствуют только структуры следования. Программы такого типа в практике встречаются редко – только для расчетов по каким-либо формулам. При этом в программе должны встречаться блоки ввода исходных данных, блоки вычислений выражений и блоки вывода результатов расчетов.

2. В работе, в целях упрощения программы, предлагается задать исходные данные с помощью оператора присваивания. Так как расчеты по формуле выполняются также с помощью операторов присваивания, в данной работе в выполняемом блоке будут только операторы присваивания и один оператор вывода результата на экран.

Запись оператора присваивания во всех случаях выполняется в виде:

<переменная> <символ присваивания> <значение выражения>;.

Следует помнить, что присваивание выполняется справа налево. Примеры:

```
Summa := 0;   {Обнуление переменной Summa}
A := B;   {Значение переменной B копируется в ячейку
      (переменную) A}
I := I+1;              {Увеличение значения
                    счетчика I на единицу}
Gip:=Sqrt(Sqr(X)+Sqr(Y)); {расчет гипотенузы по
                    величинам катетов}
```

Оператор вывода на экран в простейшем случае выглядит так:

Writeln(<список объектов вывода>);

или

Write(<список объектов вывода>);

Отличие первого варианта от второго в том, что после вывода первым оператором курсор переводится на новую строку, и следующий вывод будет выполняться в другой строке экрана.

Список объектов вывода представляет собой перечень имен переменных, строковых констант и выражений, разделенных запятыми. За каждым элементом списка может следовать формат вывода в виде одного или двух целых чисел, отделенных от элемента двоеточием. Первое число указывает, сколько позиций выделяется для выводимого значения. Если при этом значение содержит меньше символов, оно дополняется слева пробелами; если значение не помещается в отведенное место, то предлагаемый формат вывода игнорируется, а значение округляется по правилам округления.

Второе число используется только при выводе вещественных чисел и указывает, сколько дробных цифр выводить после десятичной точки. При этом число выводится в форме с фиксированной точкой (без десятичного порядка). Если второго числа в формате нет, вещественное значение выводится в экспоненциальной форме.

Примеры операторов вывода:

```
Writeln('Сколько будет чисел?'); {запрос перед вводом количества чисел}
Write(Y, Cos(Y)+1.5);{вывод значения переменной Y и значения зависящего от него выражения}
Writeln('Максимальное - ',M,' по порядку число');
Writeln(B,B:5,B:12,B:8:2,B:10:4);  {при B=12.345 будет выведена следующая строка: }
 1.23450000000012E+01 1.2E+01 1.23E+01   12.35   12.3450
```

Видно, что при выводе вещественные числа отделяются друг от друга пробелом (или знаком "–") и округляются, если не помещаются в отводимое для них поле.

Следует помнить, что целые числа при выводе без формата пишутся подряд, без пробелов, например, если K=12, L=34, а M= -5:

```
Writeln(K,L,M);   { получим результат в виде: }
1234–5
```

3. Описание переменных с десятичной точкой производится с помощью стандартного типа данных Real.

Примеры операторов описаний:

```
VAR
  A,B,C :real;
  X1,X2 :real;
VAR R,D:  real;   {раздел описаний переменных может
                   встречаться несколько раз}
```

При записи выражений на языке Паскаль нужно помнить, что написанное выражение будет выполняться слева направо, если позволяет приоритет соседних операций и отсутствуют скобки. Знаки операций для числовых выражений и их приоритеты приведены в табл. 1.

Таблица 1. Арифметические операции Паскаля

№ п/п	Название операции	Знак	Тип		Приоритет
			операндов	результата	
1	Умножение	*	Числовые	Как операнды	2
2	Деление	/	Числовые	Вещественный	2
3	Целочисленное деление	div	Целочисленные	Целочисленные	2
4	Остаток целочисленного деления	mod	Целочисленные	Целочисленные	2
5	Сложение	+	Числовые	Как операнды	3
6	Вычитание	-	Числовые	Как операнды	3

Обращение к функциям имеет более высокий приоритет (1-й), а скобки определяются как имеющие наивысший приоритет (0-й). Если в выражении соседние операции имеют разный приоритет, сначала выполняется операция с более высоким приоритетом. Примеры этого приведены в табл.2

Таблица 2. Запись математических выражений на Паскале

Вид математического выражения	Запись на Паскале	Порядок вычислений
$3{,}5\cdot10^{-3} - 2A$	*3.5e-3 - 2.0*A*	* , -
$\frac{A+B}{C-D}$	*(A+B)/(C-D)*	+ , - , /
$\frac{A\cdot B}{C\cdot D}$	*A*B/C/D*	*, /, /
$\sin X^2 + \sin^2 X$	*sin(X*X)+sqr(sin(X))*	* , *sin* , *sin* , *sqr* , +

В программе на Паскале можно пользоваться стандартной константой, соответствующей числу Pi (3.1415926...). Ее обозначение в программе – Pi, и при ее использовании нельзя описывать и применять другую переменную с таким же именем.

При работе в Турбо-Паскале (версии языка Паскаль американской компании Borland), можно пользоваться стандартными математическими функциями, имена которых приведены в табл. 3.

Таблица 3. Математические функции в Турбо – Паскале

Назначение функции	Имя функции	Тип	
		аргументов	результата
Абсолютное значение аргумента (модуль)	abs(X)	числовой	как у аргумента
Арктангенс аргумента (в радианах)	arctan(X)	Вещественный	Вещественный
Косинус (аргумент в радианах)	cos(X)	Вещественный	Вещественный
Экспонента X (e в степени X)	exp(X)	Вещественный	Вещественный
Дробная часть вещественного аргумента	frac(X)	Вещественный	Вещественный

Окончание табл.3

Назначение функции	Имя функции	Тип	
		Аргументов	результата
Целая часть вещественного аргумента	int(X)	Вещественный	Вещественный
Натуральный логарифм вещественного аргумента	ln(X)	Вещественный	Вещественный
Синус (аргумент в радианах)	sin(X)	Вещественный	Вещественный
Квадрат аргумента	sqr(X)	Числовой	Как у аргумента
Квадратный корень вещественного аргумента	sqrt(X)	Вещественный	Вещественный

Для применения других математических функций необходимо выражать их через приведенные в табл. 3, учитывая, что:

$\arcsin(X) = \arctan\frac{X}{\sqrt{1-X\cdot X}}$; $\sqrt[3]{X} = X^{1/3} = \exp(1.0/3.0\cdot\ln(X))$;

$\log_{10}(X) = \ln(X)/\ln(10.0)$; $X^Y = \exp(Y*\ln(X))$.

В качестве аргумента может выступать константа, имя переменной или выражение. Во всех случаях аргумент функции должен быть заключен в круглые скобки.

Разбор контрольного варианта

Задание

С помощью операторов присваивания задать значения аргументов, входящих в выражение, вычислить его и, присвоив полученное значение переменной *X*, вывести результат на экран

$$\sqrt{C^{D/3} - 0.5 \cdot A^{3/2} + \exp(A^{3/2} \cdot \frac{C+D}{2A})}$$

$A=10^{-2}$; $C=10^{2}$; $D=-2.5$.

Кроме имен переменных, входящих в состав выражения, нужно использовать имена встроенных функций: квадратного корня, экспоненты и натурального логарифма.

Таблица 4. Таблица идентификаторов

Имя	Тип	Размер, байт	Назначение
X	Вещественное.	6	Результат (выражения)
A	Вещественное	6	Аргумент
C	Вещественное	6	Аргумент
D	Вещественное	6	Аргумент
R	Вещественное	6	Рабочая переменная
sqrt	Вещ.функция	–	Вычисление квадратного корня
exp	Вещ.функция	–	Вычисление экспоненты
ln	Вещ.функция	–	Вычисление натурального логарифма

Блок-схема алгоритма

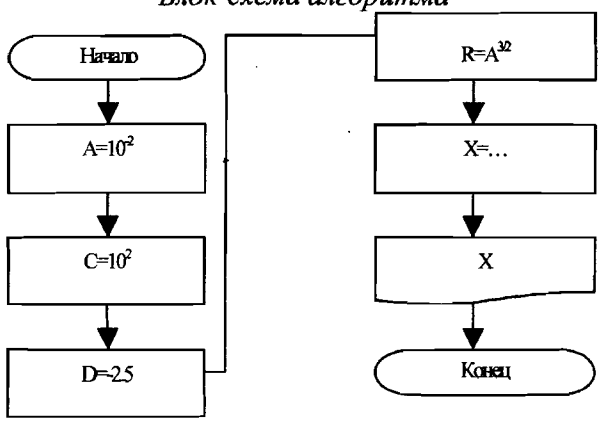


Рис.1. Алгоритм программы 26 – го варианта

Текст программы

```
PROGRAM Lab_1;
{
  Лабораторная работа N 1
    Вариант N 26
  П.Г.Петров, ст. гр. 990
}
VAR
   A,C,D,X,R   :real;
BEGIN
  A:=1e-2;
  C:=1e2;
  D:=-2.5;
  R:=Exp(1.5*Ln(A));
  X:=Sqrt( Exp(D/3.0*Ln(C)) - 0.5*R +
        Exp(R*(C+D)/2.0/A) );
  Writeln(' X= ',X:12);
END.
```

Результат:

```
X= 1.14453E+01
```

Варианты заданий

Таблица 5. Исходные данные к лабораторной работе №1

№ вар.	Формула для вычисления	A	B	C	D	Результат
1	$\sqrt{\frac{A}{2\pi(B-\sqrt{B^2-C^2})}}+\sin D$	10^5	5	2	2.5	1.95862E+2
2	$\sqrt{\frac{A/}{(2D*\ln\frac{C}{B})}}-\frac{3.5}{C}+\ln\lvert D\rvert$	10^4	10	0.1	-3	-1.48774E+1
3	$\sqrt{\frac{A}{(2\pi D(\ln\frac{88D}{B}-1.75))}}-3.5C$	10^4	10	0.2	3	1.79615E+1
4	$\sqrt{A\cdot(3D+9B+10C)/(25\pi D^2)}$	10^{-2}	-1.5	4.1	-3	1.61778E-2
5	$\sqrt{\frac{\ln A\cdot(B+C)}{4D\cdot A\cdot(B-C)}}-\exp(\frac{\sin B}{\cos C})$	10^1	-1.7	3.9	-3	-3.83304E+0
6	$\left(\frac{A}{B(C+D)\ln(A\frac{C+D}{C-D})}\right)^{3/5}$	10^3	3.5	4.1	-3	1.06442E+1
7	$\sqrt{A\cdot B\frac{(1+C\cdot\exp(D/(A\cdot B)))}{4\pi D}}$	10^1	-0.5	1.1	-1	9.65643E-1
8	$1.25\left(\frac{5.98A}{\exp\frac{B(C+1.41)^{0.5}}{87}}-D\right)$	10^2	-20.5	5.1	-1.5	1.36556E+3
9	$0.25\left(\frac{4D+A}{\exp\frac{B\sqrt{C}}{60}}-2.1D\right)$	10^{-1}	2.5	5.1	-1.5	-5.55037E-1
10	$0.25\left(\frac{\sin A}{(5.21e-6)\cdot\exp B}-4\sqrt{C\ln C}\right)-\pi D^2$	10^{-1}	1.2	5.1	2.05	1.42678E+3
11	$6.28-\frac{\sin(\ln(A)+\cos(\lg(B)}{34.2\cdot10^{-3}\exp(\sqrt[4]{C})}$	10^3	12	7.21	-	2.79759E-1
12	$\sqrt{\frac{10A+2\pi\cdot D}{1.1+\sqrt{\frac{13B}{0.5+\sqrt{\cos C}}}}}$	10	1.3	0.1	-.05	4.66048E+0
13	$7.7\cdot10^{-5}\cdot\sqrt{5.5\cdot A-\frac{2\sin(A+B)}{1-\cos(\ln C)}+\frac{\pi}{D}}$	10^{-2}	1.39	3.1	0.55	1.39860E-4

Окончание табл. 5

№ вар.	Формула для вычисления	A	B	C	D	Результат
14	$12A+\sqrt{7.41\left(B+\sin\left(\frac{C}{4}\right)\right)}-0.803\left(B\cdot\cos\frac{D}{3}+\sqrt{A}\right)$	10^{-3}	21.39	23.1	-0.12	-4.73017E+0
15	$\frac{A^B+B^A\ln C-C\lg A}{2B+D}$	10^{-1}	2.1	0.1	-3.12	-2.24257E+0
16	$\frac{4\pi}{3}A^3-\frac{2.1B\cdot10^{C+1}}{C+1-D\cdot\exp A}+\sqrt{C+1}$	10^{-3}	-2.1	1.1	-3.12	1.07743E+2
17	$\sqrt{B^{1/3}+2.4A^{3/13}}+\sin(A^{3/13}\cdot\frac{C-D}{2\pi})$	10^4	122.2	1.1	-3.12	4.39587E+0
18	$\frac{A}{B}\sqrt{\frac{(C-1)^2}{5.4B}+\frac{0.015(C-1)}{5.4A}-1+C}$	10^3	33.3	2.1	-	3.15920E+1
19	$\sqrt{\frac{A\ln D}{2-\pi+\ln D\cdot(\cos(\ln D-31\cdot10^{-3}))}}$	-10^3	-	-	10	2.96095E+1
20	$\sqrt{A\cdot(3\sin D-9\cos\lvert B\rvert+10tgC)/(25\cdot\pi\cdot D^2)}$	-10^4	0.2	-0.5	3	5.26688E-1
21	$\sqrt{\frac{A(B+\cos C)-0.3C}{4D\cdot B\cdot\exp(B+\cos C)}}$	10^4	7.7	-0.9	0.77	9.38646E-1
22	$\left(\frac{\pi A(C-D)}{B(C+D)\cdot\ln(B\cdot\frac{C+D}{C-D})}\right)^{3/8}\cdot\sin(C-D)$	10^3	-0.88	0.9	1.77	-1.08136E+1
23	$10^{-7}\left(\frac{5.98\cdot\exp(C+1.4)}{0.5+B}-\ln(C+1.4)\right)\cdot\cos\frac{A(C+1.4)}{1.81}$	10^{-1}	-0.33	2.2	-	1.28586E-4
24	$1.5\left(\frac{A/3}{\sin\frac{B\sqrt{\lvert C\rvert}}{A/3}}-\frac{3D}{A}\right)$	10^2	-0.33	-3.3	10	-2.78081E+3
25	$10^6\frac{tgA}{3.1\cdot10^6}-\sqrt[3]{\lvert C\ln\lvert C\rvert\rvert}-\pi\cdot B\cdot D^3$	10^{-1}	-0.83	-4.4	1.4	5.31933E+0
26	$\sqrt{C^{D/3}-0.5\cdot A^{3/2}+\exp(A^{1/2}\cdot\frac{C+D}{2A})}$	10^{-2}	-	10^2	-2.5	1.14453E+1

Лабораторная работа № 2.
Программирование алгоритмов с ветвлениями

Вопросы, изучаемые в работе

- Построение простейшей программы с ветвлениями.
- Изучение условных операторов.
- Использование именованных констант.
- Использование операторов ввода для исходных данных.

Задание (общее ко всем вариантам).

Написать программу вычисления и вывода на экран значения функции F по значениям одного или нескольких аргументов, величины которых вводятся с клавиатуры операторами ввода. Результат вывести на экран. Проверить программу по контрольным данным, заданным в варианте.

Оформить отчет по лабораторной работе по образцу первой работы.

Требования к программе

- Программа должна содержать комментарий по форме, указанной в работе № 1.
- Константа, встречающаяся в задании два или более раз, должна быть использована в программе в форме именованной константы.
- Значения аргументов вводятся с клавиатуры, перед вводом должен стоять оператор запроса аргументов.
- Проверку программы выполнить для всех ветвей алгоритма.
- При выводе результата одновременно выводить значения аргументов.
- Вывод результата выполнять по формату :8:4.

Общие пояснения

1.Алгоритмы с ветвлениями подразумевают, что в них существует больше одного пути, по которому можно пройти от начала к концу. Наличие параллельных ветвей алгоритма

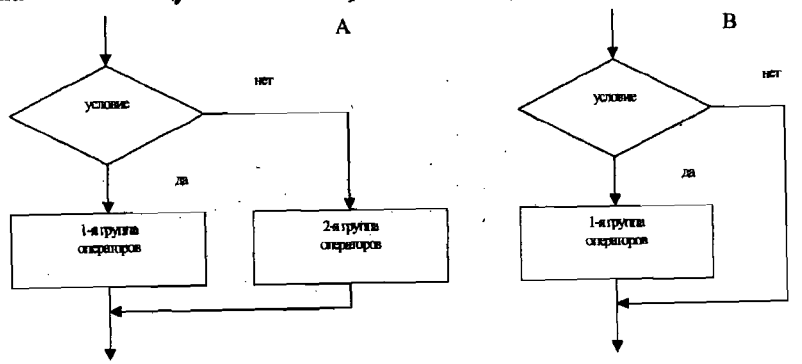


Рис. 2. Структуры А – полного и В – неполного ветвлений

осложняет тестирование программ, так как требуется задать несколько вариантов исходных данных, чтобы отработали все ветви алгоритма.

Ветвящиеся алгоритмы могут быть построены как из стандартных структур ветвления (А), так и из неполных (В).

В работе (как и вообще при программировании на языке Паскаль) предпочтительней использовать конструкцию типа (А), так как она отвечает требованиям структурного программирования.

2. Реализация структур ветвления на Паскале осуществляется с помощью условных операторов "if … then … else…".

Таблица 6. Запись "if" операторов на Паскале

Форма (А)	Форма (В) с операторами по "да"

if <услов.истинно> then
begin
<1 группа операторов>
end
else
begin
<2 группа операторов>
end;
	if <условие истинно>
	then
	begin
	<1 группа операторов>
	end;

	Форма (В) с операторами по "нет" :
	if <условие истинно> then goto Met; <1 группа операторов>; Met: . . .

Примеры написания таких операторов:

```
if A>0 then Y:=sin(X) else Y:=cos(X);
if (A+B > C) and (B < 0) then  { если требуется проверка}
 Writeln('1 группа операторов')        { нескольких условий,   }
else                                   { каждое отношение следует}
 Writeln('2 группа операторов');       { заключать в скобки    }
if  Ft then     {здесь Ft – логическая переменная,}
begin        {если Ft равно TRUE выполнится этот блок}
 Writeln(' При таких данных решения нет');
 Ft:=FALSE;
end;
```

Если в качестве оператора одной из ветвей используется условный оператор, то можно выбирать один из трех возможных путей. Вообще, количество "if"-операторов должно быть на единицу меньше, чем возможных ветвей алгоритма. Например, если нужно задать Y=-1, при X<0, Y=0 при X=0 и Y=1 при X>0, такой алгоритм и соответствующий ему текст на Паскале будут выглядеть (рис.3):

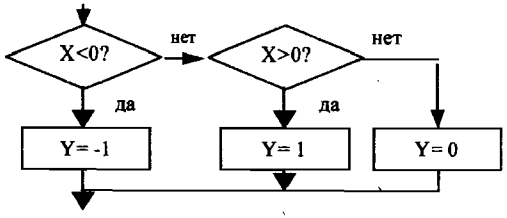


Рис. 3. Вложенный условный оператор

```
If X<0 then Y=-1
else
if Y>0 then Y=1
 else  Y=0;
```

Условиями, определяющими какую ветвь алгоритма выполнять, являются логические выражения, принимающие значение TRUE или FALSE. В качестве таких выражений часто используются отношения между двумя однотипными данными. Турбо-Паскаль разрешает сравнивать вещественные и целочисленные данные, строчные и символьные данные.

Таблица 7. Операции отношений

№ п/п	Операция	Знак операции
1	Равно	=
2	Не равно	<>
3	Больше	>
4	Больше или равно (не меньше)	>=
5	Меньше	<
6	Меньше или равно (не больше)	<=

В качестве операций отношений для упорядоченных типов данных можно использовать шесть видов операций, приведенных в табл. 7; для неупорядоченных типов разрешены только первые две операции. У всех одинаковый приоритет, причем он ниже, чем

приоритеты любых других операций (арифметических и прочих), а результат всегда имеет логическое значение.

3. В программе на Паскале можно пользоваться константами, которые имеют не только значение, но и имя. Такие константы должны быть описаны в блоке описаний, в специальном разделе описаний констант. Для таких именованных констант можно задать их значения в начале программы, где их можно, при необходимости, изменить в одном месте. Пример раздела описаний констант:

```
CONST
  MAXBALL = 5;
  MINBALL = 2;
  ERR = ' Ошибка в программе';
  ABSNUL = -273.16;
```

В дальнейшем можно всюду вместо числового значения -273.16 использовать имя ABSNUL и т.д.

Существует ряд констант, которыми можно пользоваться без их описания. Некоторые из них приведены в табл. 8.

Таблица 8. Стандартные константы Турбо-Паскаля

Имя	Тип	Значение	Назначение
TRUE	boolean	True	"истина"
FALSE	boolean	False	"ложь"
MAXINT	integer	+32767	Максимальное целое
MAXLONGINT	longint	+2147483647	Максимальное длинное целое
PI	double	3.14159265358...	Число π

4. В процессе работы программа пользуется данными, которые берет из ячеек памяти. Каким же образом эти значения попадают в эти ячейки? Существуют три возможности. Во-первых, значения могут быть занесены в некоторые переменные в самый начальный момент при загрузке программы в память. Такие переменные называются типизированными константами (хотя по сути их правильнее называть инициализированными переменными). Во-вторых, переменная (ячейка памяти) может получить значение при выполнении оператора присваивания. Наконец, в переменную можно ввести значение с помощью процедуры ввода данных с

внешнего устройства.

Только последний способ позволяет одной и той же программе обрабатывать различные наборы исходных данных. Если в программе нет операторов ввода, она при всяком запуске будет выполнять один и тот же расчет.

Оператор ввода (а вернее, процедура ввода) может вводить данные в оперативную память или из файла, или с клавиатуры. При вводе с клавиатуры процедура имеет вид:

Read(<список переменных>); или

Readln(<список переменных>);

где список переменных представляет собой перечень имен переменных через запятую, в которые заносятся вводимые значения.

Очевидно, что список значений и список имен должны соответствовать друг другу по типам и порядку следования элементов списков. Отличие в этих процедурах проявляется только при вводе данных из файла. Оно заключается в том, что при втором варианте после ввода выполняется переход на новую запись файла, даже если в текущей записи данные не кончились.

Примеры:

```
Read(N);      {программа ждет, пока не будет набрано число на клавиатуре и не нажата клавиша <Enter>, после чего переменная с именем N получит набранное значение}
Read(A,B,C);  {необходимо набрать через пробел три числа и нажать <Enter>, первое попадет в ячейку с именем A и т.д.}
```

Нельзя в списке имен писать константы или выражения.

Если в программе требуется выполнить ввод данных с клавиатуры, предварительно следует предусмотреть команды вывода на экран запроса, какие параметры и в каком порядке пользователь должен вводить, например:

```
Writeln('задай коэффициенты уравнения: A,B,C');
Readln(A,B,C);
```

или

```
Writeln('Сколько вариантов будем считать?');
Readln(N);
```

Разбор контрольного варианта

Задание

Написать программу вычисления и вывода на экран (по формату :8:4) значения функции по значениям аргументов A и B, величины которых вводятся с клавиатуры операторами ввода. Проверить ее работу для каждой ветви алгоритма заданием соответствующих исходных данных, которые заданы в табл. 9. В отчете обязательно указать, как распределяется память при выполнении задания (табл. 10). Варианты заданий приведены в табл.11.

Таблица 9. Данные задания 26-го варианта

Вид функции	При условии	Данные для проверки		
		A	B	F
$F = \sin(A+B)+1/(A+B)$	$A+B>2.13$	3.2	0.68	-0.4154
$F = \cos A - \ln(-A-B)$	$A+B<2.13$	0.34	-3.58	-0.2328
$F = \exp(A+B)/2.13$	$A+B=2.13$	2	0.13	3.9506

Таблица 10. Таблица распределения памяти

Имя	Тип	Размер, байт	Назначение
F	Веществ.	6	Результат
A	Веществ.	6	Аргумент
B	Веществ.	6	Аргумент
C	Веществ.константа		2.13
R	Веществ.	6	Рабочая переменная
sin	Веществ.функция		Вычисление синуса
cos	Веществ.функция		Вычисление косинуса
exp	Веществ.функция		Вычисление экспоненты
Ln	Веществ.функция		Вычисление натурального логарифма
Lab_2	Имя программы		Вычисление заданной функции

Блок-схема алгоритма

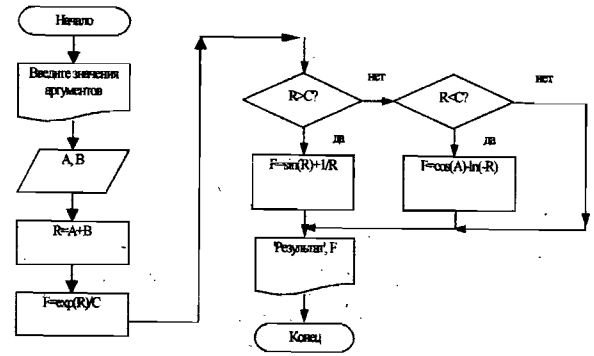


Рис. 4. Алгоритм 26-го варианта

Текст программы

```
PROGRAM Lab_2;
{    Лабораторная работа N 2 Вариант N 26
  П.Г.Петров, ст. гр. 990  }
CONST
 C=2.13;
VAR
 A,B,F,R   :real;
BEGIN
 Writeln('Значения аргументов A и B ?');
 Readln(A,B);
 R:=A+B;
 F:= exp(R)/C;
 if R>C then
   F:= Sin(R)+1.0/R
 else
   if R<C then
     F:= cos(A)-ln(-R);
 Writeln(' Рез-т: ',F:8:4);
END.
```

Получены результаты по заданным наборам данных:

```
Рез-т:  -0.4154
Рез-т:   0.2328
```

Варианты заданий

Таблица 11. Варианты заданий лабораторной работы № 2

№ вар.	Вид функции	При условии	Данные для проверки		
			A	B	Результат
1	$F = \exp(A/2) + \sqrt{-A}$	$A < -1.25$	- 4.0	0	2.1353
	$F = \lvert 1.25 - A \rvert + \lg B$	$A \geq -1.25$ и $B > 1$	- 1.0	100	4.2500
	$F = \sin(A \cdot B)$	*в остальных случаях*	4.52	0.25	-0.9983
2	$F = B\sin A + 7.04A$	$B < 7.04$	0.77	1.99	6.8061
	$F = A\sin B + A\cos B$	$B > 7.04$	2.88	10	-3.9833
	$F = A\arcsin(B/10)$	*в остальных случаях*	1.01	7.04	0.7888
3	$F = tg(A-1.5) - 1.5/B$	$\lvert A-1.5 \rvert < 1.5$ и $\lvert B \rvert > 0.1$	2.10	-1.2	1.9341
	$F = \exp(B/(A-1.5))$	$\lvert A-1.5 \rvert > 1.5$ и $\lvert B \rvert < 0.1$	10.2	20.1	20.8130
	$F = \sin(A-1.5) + 1.5B$	*в остальных случаях*	3.0	1.0	2.497
4	$F = \lvert 1.5 \cdot 10^{-2} \cdot A + B \rvert$	$\lvert A \rvert < 1.5 \cdot 10^{-2}$ и $B < 1$	0.01	0.5	0.5001
	$F = \ln B - \exp(1.5e - 2 \cdot A)$	$\lvert A \rvert < 1.5 \cdot 10^{-2}$ и $B \geq 1$	0.01	2.0	-0.3070
	$F = B/(A \cdot arctgA)$	*в остальных случаях*	1.11	-2.22	-2.3881
5	$F = \sqrt{0.13456A} + B - 0.13456$	$A > 0.13456$	65.43	1.33	4.1626
	$F = \exp(A) - 0.13456B$	$\lvert A \rvert \leq 0.13456$	0.11	10	-0.2293
	$F = \lvert A - B \rvert / (0.13456 + \lvert A \rvert)$	*в остальных случаях*	-2.13	-3.13	0.4416
6	$F = 2.07\sqrt{A \cdot B} + \lvert A \cdot B \rvert$	$A \cdot B > 2.07$	-2	-2	-0.14
	$F = 1/(2.07A \cdot B)$	$A \cdot B < -2.07$	-1.0	3	-0.1610
	$F = \dfrac{\pi A - 2.07\sin(A-B) + 1}{2.07 + \lvert A \rvert}$	*в остальных случаях*	1	1.0	1.3491
7	$F = \lvert A \rvert^{B} - B/A$	$\lvert A \rvert > 0.333$ и $\lvert B \rvert < 2$	1	0.55	0.4500
	$F = e^{-B} \cdot 0.333A$	$\lvert A \rvert > 0.333$ и $\lvert B \rvert \geq 2$	2	5.45	0.0211
	$F = \cos(A \cdot B) - 0.333A$	*в остальных случаях*	0.11	10	0.4170
8	$F = tgB + \sin A$	$\lvert B \rvert < 0.7788$	2.22	0.5	1.3429
	$F = \sin(A \cdot B) - \cos(A \cdot B)$	$A < 0.7788$ и $\lvert B \rvert \geq 0.7788$	0.5	2.22	0.451
	$F = \pi\sqrt{\lvert B \cdot A \rvert + 0.7788}$	*в остальных случаях*	1	0.1	0.9418
9	$F = (A+B)/(A-B-0.3456)$	$A - B > 0.3456$	1.5	0.5	3.0562
	$F = \sin(\pi/5 - A + B)$	$A - B < -0.3456$	0.5	1.5	0.9983
	$F = \cos(B - A - 0.3456)$	*в остальных случаях*	0.2	0.01	0.9374

Продолжение табл.11

№ вар.	Вид функции	При условии	Данные для проверки		
			A	B	Результат
10	$F = \sqrt{\lvert 462 \cdot 10^{-3} - A \rvert}$	$\lvert B \rvert \geq 462 \cdot 10^{-3}$ и $A > 0$	-1	1.0	1.2091
	$F = \sin(A \cdot B) - 4.62e - 1$	$\lvert B \rvert < 462 \cdot 10^{-3}$ и $A < -2$	-3	0	-0.4620
	$F = A - \sqrt{0.462 - \lvert B \rvert}$	*в остальных случаях*	-1.0	0.0	-1.6797
11	$F = 4.777A^{B/4.777}$	$A > 0$ и $B > 0$	1.1	2.2	4.9914
	$F = \sin(A - 4.777B) - 4.777B$	$A > 0$ и $B \leq 0$	3.3	-1.2	6.1148
	$F = B - \lvert A + 4.777 \cdot B \rvert$	*в остальных случаях*	0	-1	-5.7770
12	$F = \lvert A \rvert^{0.234} + B/0.234$	$A \leq -0.234$ и $B < 0$	-1	-2	-7.547
	$F = \cos(A + 0.234) \cdot \sin(A + 0.234)$	$B \geq 0$	0.0	1.0	0.2256
	$F = \dfrac{\lvert A + B \rvert}{0.7(A + 0.234)}$	$A > -0.234$ и $B < 0$	0	- 0.23	1.4286
13	$F = 1.123arctgA + B/1.123$	$\lvert B \rvert < 1.123$	1	-1	-0.0085
	$F = \exp(1.123A)/B$	$\lvert B \rvert \geq 1.123$ и $A < 3$	2.5	2.1	7.8897
	$F = 1.123\sin A - \exp(-B)$	*в остальных случаях*	3.14	2	-0.1335
14	$F = A^{0.5555} - 0.5555B$	$A > 0$ и $\lvert B \rvert \leq 0.5555$	1	-0.1	1.0556
	$F = \sqrt{A}/B - 0.5555B$	$A > 0$ и $\lvert B \rvert > 0.5555$	4	2	-0.1110
	$f = 0.5555\exp A \cdot \sin B$	$A \leq 0$	-1	0	0.0000
15	$F = arctg(0.3456B) - A \cdot B$	$\lvert A \cdot B \rvert < 0.3456$	1	-0.2	0.131
	$F = \lg\lvert A \cdot B \rvert + \sqrt{A \cdot B}$	$\lvert A \cdot B \rvert \geq 0.3456$ и $A \cdot B > 0$	2.5	1.5	2.5105
	$F = \sin B - \cos(0.3456A \cdot B)$	*в остальных случаях*	1.5	-1.5	-1.7101
16	$F = \lg A - \sin B + 0.444$	$A > 0.444$	1	0.5	-0.0354
	$F = \exp(0.444A) + \cos B$	$A < -0.444$	-1	2.3	-0.0248
	$F = 0.444\ln\lvert B \rvert$	*в остальных случаях*	0	9.5	0.9996
17	$F = A - (A-B)^{0.1357}$	$A > B$	3.5	2.5	2.5
	$F = 0.1357\cos^2(A - B)$	$A \leq B$ и $B < 0$	-3.5	-2.5	0,0396
	$F = \sin^2(A - B - 0.1357)$	*в остальных случаях*	1	2	0.8223
18	$F = \sqrt{A+B} - 1.4973B$	$A + B > 0$	2	-1	2.4973
	$F = (A^{1.4973} - B)(A + B)$	$A + B \leq 0$ и $A > 0$	1	-2	-3.0000
	$F = \lvert A + B \rvert / (1.4973 - A)$	*в остальных случаях*	-1.5	0.5	0.3336

Окончание табл.11

№ вар.	Вид функции	При условии	Данные для проверки		
			A	B	Результат
19	$F = 2.8765\ln(A \cdot B)$	$A + B > 0$	1.5	2	3.1602
	$F = \cos(2.8765A) + B$	$A + B \le 0$ *и* $B > -2.8765$	-1.5	-2	-2.3872
	$F = 2.8765 / B$	*в остальных случаях*	0	-5.8	-0.4959
20	$F = A^{3.1212} + 3.1212B^{0.777}$	$A > 0.777$ *и* $B > 0.777$	1	1.0	4.1212
	$F = \cos(2.8765A) + B$	$A < -0.777$ *или* $B < -0.777$	-1	2	-0.2264
	$F = 2.8765 / B$	*в остальных случаях*	0.5	2	3.8982
21	$F = \lg(A - B)$	$A - 10^{-3} > B$	1.8	0.8	-0.0000
	$F = arctg\lvert A - B\rvert + 0.001$	$A < B$ *и* $A > 0$	0.5	1.7	0.8771
	$F = A + 0.001(A - B)$	*в остальных случаях*	-0.2	100	-0.3002
22	$F = 4.6241\sqrt{A + B}$	$A + B > 4.6241$	200	-89.9	48.52
	$F = 1/(4.6241 - A - B)$	$A + B < -4.6241$	-12.3	4.5	0.0805
	$F = (A + B)^5$	*в остальных случаях*	2.5	-0.5	32.0000
23	$F = A^{0.333} - B^{0.333}$	$A > 0$ *и* $B > 0$	10	20	-0.5589
	$F = \ln(A + B)$	$A \le 0$ *и* $B \le 0$ *при* $A > -B$	-10	15	1.6094
	$F = 0.333(A + B)^8$	*в остальных случаях*	3.5	-5.5	85.248
24	$F = \sin^4(1.121 \cdot A \cdot B)$	$A < 0$ *и* $B < 0$	-1.88	-0.66	0.9370
	$F = \cos(1 / A / B)$	$A > 0$ *и* $B > 0$	1.25	0.13	0.9916
	$F = 1.121 \cdot A \cdot B$	*в остальных случаях*	5.5	-0.02	-0.1233
25	$F = \ln\frac{A}{B} - \frac{A}{8.338B}$	$\frac{A}{B} > 8.338$ *и* $\lvert B\rvert > 0.1$	11.1	0.87	1.016
	$F = \exp\frac{A}{B} + 8.338B$	$\frac{A}{B} \le 8.338$ *и* $\lvert B\rvert > 0.1$	3.9	1.55	25.3045
	$F = 8.338B$	*в остальных случаях*	0	0.06	0.5003
26	$F = \sin(a + b) + 1/(A + B)$	$A + B > 2.13$	3.2	0.68	-0.4154
	$F = \cos A - \ln(-A - B)$	$A + B < 2.13$	0.34	-3.58	-0.2328
	$F = \exp(A + B)/2.13$	$A + B = 2.13$	2	0.13	3.9506

Лабораторная работа № 3.
Работа с последовательностями чисел

Вопросы, изучаемые в работе

- Построение программы циклической структуры с использованием операторов арифметических и итеративных циклов, реализация последних с помощью условных переходов.
- Освоение типовых алгоритмов: вычисления суммы, произведения, поиск максимума, минимума во вводимой последовательности и их порядковых номеров.
- Использование операторов описания переменных для данных различных типов.
- Использование в программе контроля за входными данными и результатами расчета.

Задание (общее ко всем вариантам)

Составить программу обработки последовательно вводимых с клавиатуры чисел (пока не будет обработано заданное количество чисел или не сработает условие окончания ввода). Полученный результат обработки вывести на экран.

Оформить отчет по работе аналогично оформлению отчета по работе № 2. Текст программы должен быть распечатан, результаты – переписаны от руки.

Требования к программе

- Программа должна обрабатывать данные указанного типа в количестве до 100 чисел, если в программе не указано иное ограничение. Конкретное количество чисел вводить в программу с клавиатуры. Массивы не использовать.
- Все значения, на которые по смыслу накладываются ограничения, должны при вводе проверяться.
- Если в результате обработки данных результат не получен, при выводе программа должна сообщать об этом в понятной форме.
- При выводе на экран использовать длину выводимой строки не более 76 символов.
- Остальные требования – как и в предыдущих лабораторных работах.

Общие пояснения

1. Программы циклической структуры используются, когда необходимо несколько раз выполнить однотипные действия с различными данными. Если количество повторений тела цикла известно перед началом цикла, он называется арифметическим, если нет – итеративным. Для организации арифметического цикла в блок-схеме алгоритма используется блок "модификатор", а в программе – соответствующий ему оператор "for...".

Итеративный цикл строится с использованием блока "решение", в котором один из альтернативных путей представляет выход из тела цикла. В программе такой цикл может выполняться с помощью специальных операторов итеративных циклов или оператора условного перехода. В данной работе будет использоваться только "if..." оператор.

Рассмотрим сначала *арифметический цикл*. Оператор состоит из заголовка и тела цикла. Заголовок имеет вид:

for <имя параметра цикла> := <начальное значение> to <конечное значение> do

Далее идет тело цикла: простой или составной оператор (до символа ";"). Если оператор составной, он заключается в операторные скобки begin ... end;

В качестве параметра цикла можно использовать любую целочисленную переменную, в которой в это время не хранится нужное в дальнейшем значение. Эта переменная изменяется в цикле автоматически. Параметр цикла – это переменная, обычно играющая роль не только счетчика количества выполненных повторений цикла, но одновременно служащая порядковым номером обрабатываемого числа или элемента массива.

Начальное и конечное значения параметра цикла могут быть выражениями, но они вычисляются только один раз – при первом входе в цикл. В теле цикла они не должны меняться. При каждом возврате к заголовку в цикле счетчик автоматически увеличивается на единицу.

Тело цикла выполняется, пока счетчик не станет больше конечного значения (если счетчик равен конечному значению – цикл выполняется). Если требуется организовать цикл, в котором параметр уменьшается на единицу с каждым возвратом, то используют ключевое слово "downto", вместо "to".

Примеры написания оператора:

```
for i:=1 to 25 do  write('*'); {вывод строки из 25 *}
for k:=L+1 to N-1 do    { значения L и N должны   }
begin                   { быть определены до цикла }
 . . .
end;
for i:=N downto 2 do S:=S+i; {Сумма целых чисел 2..N}
```

Если нужно менять счетчик с другим шагом (не один), используют второй счетчик, изменяющийся в теле цикла по рекуррентной формуле, например j:=j+3; или вычисляемый через первый счетчик, например: j:= 2+(i-1)*3;

Итеративные циклы для завершения требуют проверки такого условия, результат которого может измениться в процессе выполнения тела цикла. По выполнении этого условия осуществляется переход на оператор, следующий за телом цикла. Возврат на начало тела цикла в этом случае выполняется оператором безусловного перехода. В программе при такой организации цикла требуется иметь две метки:

```
Start:
 . . .
 if A=Priznak then goto Finish;
 . . .
 goto Start;
Finish: . . .
```

Часто проверяют условие продолжения цикла и по его выполнению возвращаются на начало тела цикла, иначе цикл заканчивается. В этом случае достаточно обычно одной метки:

```
Start:
 . . .
 if A<>Priznak then goto Start;
 . . .
```

Тело итеративного цикла при таких способах построения в скобки begin ... end заключать не обязательно.

2. Циклические процессы почти всегда требуют некоторых подготовительных действий, выполняемых до начала цикла. Это связано с использованием в теле цикла так называемых "рекуррентных соотношений", в которых некоторая переменная вычисляется с использованием своего старого значения, например: N:=N-1; P:=-2*P/i; S:=S+A и т.д.

Все такие переменные перед циклом должны получить определенные значения, чтобы правильно вычисляться внутри цикла. Например, при вычислении суммы последовательно вводимых слагаемых ячейка (переменная) для суммы должна обнуляться; переменная для накопления произведения делается равной единице (или первому сомножителю).

Поиск максимумов (минимумов) или их порядковых номеров в рядах значений также выполняется по рекуррентным зависимостям (если новая величина больше максимума, сделаем максимум равным...) – сперва используется старое значение максимума, чтобы создать новое значение.

При поисках максимума можно использовать два варианта начального задания: либо в качестве максимума берется первое рассматриваемое значение (и цикл выполняется, начиная со второго числа), либо в качестве начального задается фиктивное значение – гарантированно меньшее, чем любое число сравниваемого ряда (т.е. теоретически минимальное возможное число). При этом циклическая обработка одинаково выполняется для всех чисел, начиная с первого.

3. Если в программе требуется обрабатывать нечисловые данные, для них можно заводить переменные нечисловых типов, которые также необходимо включить в блок описаний. Для символьных переменных используется описатель char, а для логических – описатель boolean. Например:

```
VAR
 i,j :integer;
 A,S :real;
 C,Sim :char;
 Q,Priz:boolean;
...
```

Символьные переменные можно сравнивать с символьными константами и между собой с помощью всех операций отношений (=, <, > и т.д.), например:

```
 if Sim > 'A' then ... или
 if C = Sim then ...
```

Логические переменные можно использовать в условных операторах, причем не нужно записывать if Q=TRUE then..., а просто if Q then... , так как само значение Q может быть либо TRUE, либо FALSE.

Наконец, несколько замечаний по контролю за вводимыми величинами. Поскольку в программе предполагается ввод исходных данных пользователем программы путем набора значений на клавиатуре, в программе должны быть предусмотрены выводы на экран запросов на ввод того или иного параметра, нужного программе. Эти запросы представляют собой операторы вывода некоторого текста, например:

```
 Writeln(' Сколько чисел будет обрабатываться?'); или
 Writeln(' Вводи очередной сомножитель');
```

Такие запросы должны быть перед каждым оператором ввода данных с клавиатуры. Другой момент, который следует отметить, это необходимость контроля за вводимыми величинами. Недопустимое значение может не только привести к неправильному ответу, но и просто "подвесить" программу или привести к ее аварийному снятию операционной системой. Поэтому все данные, на которые по их смыслу (или по возможностям реализованного метода решения) накладываются ограничения, должны проверяться после их ввода на попадание в допустимый диапазон.

Обычно считается, что можно найти сумму нуля и большего количества слагаемых, произведение одного или более сомножителей, выбрать наибольшее или найти среднее значение из одного или нескольких чисел, вычислить факториал нуля или большего целого числа и т.д. Иногда ограничения накладываются не только снизу, но и сверху.

Предположим, допустимый диапазон для значений переменной A ограничен числами –1.7 и 15.0, то проверку на ошибочное значение можно осуществить оператором:

```
 if (A > 15.0) or (A < -1.7) then ...
```

Если такая проверка даст значение TRUE (т.е. A имеет недопустимое значение), тогда нужно выдать сообщение об этом и либо закончить выполнение программы, либо (что более разумно) вернуться к оператору запроса ввода этой величины еще раз. Так как оба эти действия следует выполнять по срабатыванию одного и того же условия, операторы, задающие эти действия, следует поместить в скобки begin ... end, например:

```
 if (A > 15.0) or (A < -1.7) then
 begin
  Writeln(' Вы задали недопустимое значение');
```

```
goto Vvod_A;
end; { здесь Vvod_A – метка перед оператором запроса на ввод параметра, соответствующего переменной A }
```

Наконец, следует помнить, что не при всяких наборах исходных данных задача может иметь решение. Если по каким-либо причинам ответ выдать невозможно, следует об этом сообщить в понятной форме, а не выводить неправильное или невозможное для ответа значение. Например, если в задаче требуется делить что-то на сумму нескольких чисел с разными знаками, то сумма может случайно оказаться равной нулю, после чего деление невозможно и решения у задачи не окажется.

Тогда выдача результата может выглядеть так:

```
...
if Sum = 0 then
Writeln(' Нет решения, так как сумма равна 0')
else
begin
 R:=.../Sum;
 Writeln(' Отношение =', R:...);
end;
```

Разбор контрольного варианта

Задание

Таблица 12. Данные к заданию 26-го варианта

№ варианта	Задание	Остановить обработку при…	Тип обрабатываемых данных
26	Нахождение номера последнего числа, превышающего значение 10.5, в последовательности вводимых произвольных чисел.	…вводе заказанного количества чисел	Вещественные

Решение задачи необходимо начать с разбора задания и выделения объектов, упоминающихся или подразумевающихся в задании(табл. 12). Для каждого найденного объекта определяется

его тип и задается имя (идентификатор) для последующего программирования. Имена, типы и назначения сводятся в таблицу идентификаторов.

Для данной задачи можно выделить: номер последнего числа, превышающего значение 10.5; количество чисел (*N*); вводимое число. Для запоминания искомого номера нужно знать порядковый номер вводимого числа. В задании оговорено, что вводимые числа – произвольные, т.е. могут быть как целыми, так и дробными, для их хранения в ЭВМ нужно иметь переменную (ячейку) вещественного типа. Текущий номер и искомый номер – величины целые и положительные. Для них можно завести или целые или беззнаковые переменные. Общее количество чисел также должно быть целым и больше нуля, иначе задача не имеет смысла (табл.13):

Таблица 13. Идентификаторы программы 26-го варианта

Имя	Тип	Размер, байт	Назначение
N	Целый	2	Количество чисел
I	Целый	2	Текущий номер
Num	Целый	2	Номер последнего числа, которое > 10.5
A	Вещественный	6	Текущее число

Исходными данными в задаче являются, во-первых, количество чисел, а во-вторых, сами числа, последовательно вводимые в переменную A. Результатом будет порядковый номер последнего из чисел, которое превышает 10.5. Возможно, что среди вводимых чисел не найдется ни одного такого числа. В этом случае в конце необходимо выдать об этом текстовое сообщение, а не номер числа.

Контроль допустимости вводимых данных необходим только для количества чисел: оно должно быть не меньше единицы.

Алгоритм задачи состоит из трех последовательных обобщенных шагов: ввода данных, определения искомого номера и вывода результата.

В части ввода данных программа должна получить от пользователя значение количества чисел (*N*). Так как ввод данных выполняет человек с помощью клавиатуры, программа должна сообщить, что от него требуется.

Всякому вводимому с клавиатуры числу должен предшествовать запрос на дисплее: что вводить и в какой форме. Как и любое значение, вводимое пользователем с клавиатуры, количество чисел (*N*) должно контролироваться на допустимость введенной величины.

Если введено недопустимое значение, требуется выполнить два действия: сообщить об этом и вернуться на запрос нового значения N.

Возврат назад возможен двумя способами: командой перехода на метку, поставленную перед оператором запроса, или оператором итеративного цикла, причем цикла с "постусловием", так как один раз цикл должен выполниться обязательно.

В данном случае используем первый способ, и потому в программе появляется еще один объект – метка. Дадим ей имя metka.

Кроме того, дадим имя нашей программе, например laborat_N_3. В результате в таблице имен добавится две строки (табл. 14):

Таблица 14. Окончание таблицы идентификаторов

Имя	Тип	Размер, байт	Назначение
Laborat_N_3	Имя программы	-	Поиск номера определенного числа
Metka	Метка	-	Возврат по ошибке ввода

Алгоритм

Раздел поиска номера обычно включает подготовку и цикл поиска. Так как количество чисел известно, используется арифметический цикл. В тело цикла входит получение очередного числа и, если нужно, запоминание его номера. Если встретится хотя бы одно число, удовлетворяющее условию превышения значения 10.5, номер будет найден, иначе нужно иметь признак, что такого числа не встретилось. В качестве такого признака можно использовать переменную *Num*, в которую перед циклом поиска заносится значение, невозможное для номера числа: например, –1. Если после цикла *Num* останется равным –1, следовательно, искомых чисел не встретилось.

Таким образом, раздел поиска включает присваивание начального значения переменной *Num* и цикл, тело которого содержит запрос ввода очередного числа, прием и занесение его в переменную для текущего числа *A*; проверку, что *A*>10.5, и если так, запоминание текущего номера в переменной *Num*. Цикл поиска нужно вести до конца, так как нас интересует последнее число, удовлетворяющее условию выбора.

По окончании раздела поиска в разделе вывода результатов печатается найденный номер *Num*, если он не равен –1, иначе выдается сообщение об отсутствии чисел, удовлетворяющих условию задачи.

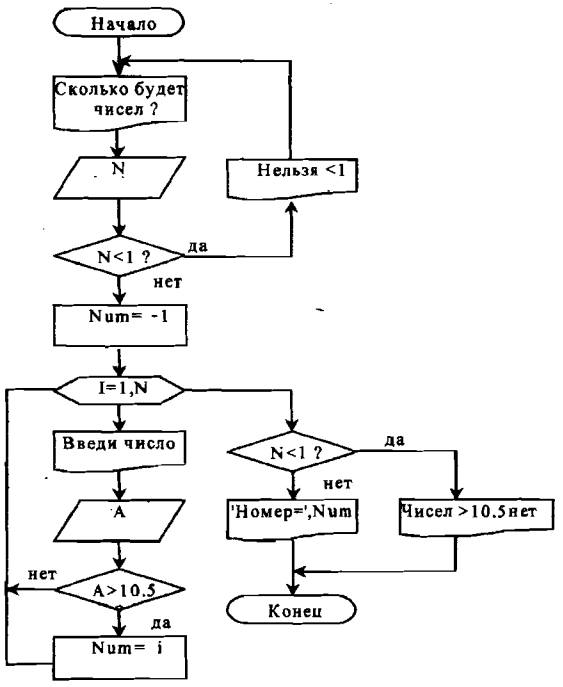


Рис. 5. Алгоритм 26-го варианта

На основании построенного алгоритма можно написать текст программы, причем таблица идентификаторов используется для

создания раздела описаний, а алгоритм – для выполняемого блока. При написании учтем требования к оформлению текста программы. Получим следующую программу на Паскале. Варианты заданий приведены в табл.15.

Текст программы

```
PROGRAM laborat_N_3;
{  Программа Лабораторной работы N 3
    Вариант N 26.
 Петров П.Г., ст. гр.990  }
VAR   N,Num,i :integer;
      A    :real;
LABEL   metka;
BEGIN  { Выполняемый блок.
     Первый раздел - ввод данных }
 metka: writeln('Сколько будет чисел?');
 readln(N);
 if N < 1 then
 begin
  writeln('Нельзя рассматривать меньше 1 числа');
  goto vvod;
 end;
   { Основной раздел - поиск номера }
 Num:= -1; {задание признака отсутствия подходящих чисел}
 for i:=1 to N do
 begin
  writeln('Введите очередное число');
  readln(A);
  if A > 10.5 then  { если число удовлетворяет
                 условию задачи }
    Num:=i;       { запоминаем его номер }
 end;
  { Раздел вывода результатов поиска }
 if Num=-1 then writeln('Подходящих чисел не было')
 else
  writeln('Номер последнего числа,>10.5 равен ',
        Num);
END.
```


Варианты заданий

Таблица 15. Варианты заданий лабораторной работы № 3

№ вар.	Задание	Остановить обработку при…	Тип обраб. данных
1	Вычисление произведения последовательно вводимых чисел	…превышении абсолютной величины произведения 1000.0, или при вводе числа 0	Веществ.
2	Вычисление суммы только четных чисел из последовательно вводимых чисел	…вводе заказанного количества чисел	Целые
3	Нахождение номера минимального значения в последовательности вводимых чисел	…вводе признака конца – предварительно введенного пользователем числа	Целые
4	Вычисление суммы только положительных чисел из последовательно вводимых чисел	…превышении суммы 100.0	Веществ.
5	Нахождение максимального значения в последовательности вводимых чисел. При поиске пропускать числа из диапазона от 10.1 до 50.1	…вводе заказанного количества чисел	Веществ.
6	Вычисление суммы последовательно вводимых чисел	…вводе признака конца – значения из заранее выбранного пользователем диапазона	Веществ.

Продолжение табл. 15

№ вар.	Задание	Остановить обработку при ...	Тип обраб. данных
7	Нахождение минимального значения среди последовательности вводимых чисел. Сам признак конца при определении минимума не учитывать	...вводе признака конца – предварительно введенного пользователем числа из диапазона -10..5	Целые
8	Вычисление произведения заданного количества последовательно вводимых чисел	...вводе заказанного количества чисел – не меньше 5 но и не более 10	Веществ.
9	Нахождение минимального значения из чисел, кратных трем, среди последовательности вводимых чисел	...вводе заказанного количества чисел	Целые
10	Вычисление суммы последовательно вводимых чисел	...превышении абсолютной величины суммы 100.1	Веществ.
11	Нахождение максимального из последовательно вводимых чисел	...вводе признака конца – предварительно выбранного пользователем числа	Целые беззнаковые
12	Вычисление произведения последовательно вводимых чисел	...вводе признака конца – числа из диапазона от -5 до 5 (включительно)	Целые

Продолжение табл. 15

№ вар.	Задание	Остановить обработку при ...	Тип обраб. данных
13	Нахождение номера максимального значения, кратного пяти, в последовательности вводимых чисел	...вводе признака конца – предварительно выбранного пользователем числа	Целые
14	Вычисление факториала вводимого числа. (Факториалом целого числа N (обозначается N!) называется произведение всех целых чисел от 1 до N. По определению, 0!=1)	...если результат не помещается в 4 байта	Длинные целые (4 байта)
15	Вычисление среднеарифметического тех из последовательно вводимых чисел, которые попадают в диапазон от 2.0 до 5.0 включительно.	...вводе заказанного количества чисел	Веществ.
16	Нахождение номера максимального отрицательного значения в последовательности произвольных вводимых чисел	...вводе признака конца – предварительно выбранного пользователем числа	Веществ.
17	Вычисление произведения только отрицательных из последовательно вводимых произвольных чисел	...вводе заказанного количества чисел	Веществ.
18	Нахождение номера наименьшего положительного числа в последовательности произвольных чисел, вводимых с клавиатуры	...вводе заказанного количества чисел	Веществ.

Продолжение табл. 15

№ вар.	Задание	Остановить обработку при ...	Тип обраб. Данных
19	Вычисление среднеарифметического последовательно вводимых чисел	...вводе третьего отрицательного значения, которое рассматривать как признака конца, а не число.	Веществ.
20	Нахождение максимального отрицательного значения в последовательности вводимых произвольных (как положительных, так и отрицательных) чисел	...вводе заказанного количества чисел	Целые
21	Вычисление среднего значения только положительных элементов среди вводимых произвольных чисел	...вводе заказанного количества чисел	Веществ.
22	Нахождение разности номеров первого и последнего отрицательных чисел в последовательности водимых чисел	...вводе заказанного количества чисел	Целые
23	Вычисление среднеарифметического только отрицательных чисел среди произвольных вводимых чисел	...вводе признака конца (выбранного пользователем произвольного числа)	Веществ.
24	Нахождение разности максимального и минимального значений в последовательности вводимых чисел	...вводе заказанного количества чисел	Целые

Окончание табл. 15

№ вар.	Задание	Остановить обработку при ...	Тип обраб. данных
25	Вычисление суммы только отрицательных чисел, абсолютная величина которых превышает 8.5, из последовательно вводимых произвольных чисел	...вводе заказанного количества чисел	Веществ.
26	Нахождение номера последнего числа, превышающего значение 10.5, в последовательности вводимых произвольных чисел	...вводе заказанного количества чисел	Веществ.

Лабораторная работа № 4
Работа с одномерными массивами
(подсчет, поиск элементов, перестановки в массиве)

Вопросы, изучаемые в работе

- Изучение простейших составных данных – одномерных массивов.
- Освоение форматного вывода одномерных массивов разных типов.
- Создание и использование в программах входных текстовых файлов с данными.
- Вывод результатов работы в выводной текстовый файл (протокол работы).
- Дальнейшее изучение основных элементарных алгоритмов.

Задание (общее ко всем вариантам)

Составить программу обработки одномерного массива заданного типа произвольной длины. Количество используемых данных должно вводиться с клавиатуры (в пределах отведенного под массив места). Значения элементов массива вводятся или с клавиатуры, или из предварительно созданного текстового файла (в соответствии с заданием). Исходный массив и результаты его

обработки выводить в выводной текстовый файл.

Оформить отчет по работе, аналогично отчету по работе № 3, но в печатном виде должны быть представлены текст программы и результаты работы.

Требования к программе

- Если специально не указано, что данные в массив вводятся с клавиатуры, перед запуском программы необходимо подготовить данные требуемого типа в отдельном текстовом файле, которому следует присвоить имя, совпадающее с именем файла Паскаль-программы, но с расширением ".DAT". Данные набивать, отделяя друг от друга пробелом, по 10 – 20 чисел в строке. Для символьных данных – все набивать в одной строке.
- Программа должна обрабатывать данные указанного типа в количестве до 50 чисел и до 250 символов, если количество не указано в задании. Конкретный размер массива при запуске программы вводить с клавиатуры.
- Все значения, на которые по смыслу накладываются ограничения, должны при вводе контролироваться.
- Вывод исходных данных и результатов производить в выходной текстовый файл. При выводе использовать длину выводимой строки не более 76 символов.

Содержание программы

- Заголовок программы с комментарием.
- Описание переменных и массива.
- Описание меток.
- Ввод исходных данных с клавиатуры.
- Открытие входного и выводного файлов.
- Заполнение массива из файла.
- Вывод на печать введенного массива в выводной файл под заголовком "Исходные данные".
- Проведение обработки массива в соответствии с заданием.
- Вывод результатов обработки в выводной файл под заголовком "Результаты расчета".
- Закрытие всех файлов.

Общие пояснения

Рассмотрим некоторые моменты, связанные с обработкой одномерных массивов. Следует помнить, что выделение памяти под массив при его описании может производиться только с помощью констант (в явном виде или с предварительным описанием константы), например:

```
Const MIN=5;
    MAX=100;
Var  Massiv1 : array[1..110] of real;
   Massiv2 : array[MIN..MAX] of integer;
```

Кроме того, рекомендуется сначала завести свои описатели типов массивов, а затем с их помощью выделять место под конкретные переменные этих "массивных" типов, например:

```
Const  MAXI=200;
Type     IntMasMAX = array[1..MAX] of integer;  {описатель для целочисленных массивов из MAX элементов}
     RealMass = array[1..110] of real; { описатель
     для вещественных массивов из 110 элементов }
Var Massiv1,Nmbs:IntMasMAX; {завели 2 целочисл. массива}
   Massiv2,WindV:RealMass; {завели 2 веществ. массива}
```

Программы, использующие массивы, позволяют сначала ввести все необходимые данные, а только потом их использовать. Поэтому в таких программах следует предусматривать следующие шаги (в указанном порядке):

1. Ввод исходных данных. Обычно включает ввод количества элементов массива (с проверкой допустимости введенного значения) и ввод самих элементов массива в указанном количестве. Если ввод значений предусмотрен с клавиатуры, перед каждым оператором чтения должен быть запрос на ввод. Если используется ввод из файла, запросы не делаются, но перед вводом данных в массив из файла последний следует открыть для чтения стандартными процедурами Assign и Reset, а в разделе описаний переменных для файла должна быть введена переменная с помощью описателя text:

```
VAR
 Finput : text; {завели файловую переменную для набора данных текстового типа}
  . . .
BEGIN
```

```
. . .
 Writeln(' Введи длину массива');
vvod: Read(N);
 if(N<MIN) or (N>MAX) then
 Begin
  Writeln('Недопустимое количество, введите снова');
  goto vvod;
 End;
 Assign(Finput,'UMNIK4.DAT');    { связали файловую
          переменную с набором данных UMNIK4.DAT }
 Reset(Finput);          { открыли файл для чтения }
 for i:=1 to N do
  Read(Finput,Massiv2[i]);
 . . .
```

Если в задании требуется вводить данные в массив, пока не встретится определенный признак, то можно поступить двумя способами. Организовать арифметический цикл со счетчиком от одного элемента до предельно допустимой длины массива и дополнительным выходом из цикла, если встретился признак; или использовать итеративный цикл с условием продолжения, если введенное значение – не признак, и его порядковый номер меньше длины массива.

2. Вывод исходных данных. Включает обычно вывод заголовка массива, возможно с указанием его длины, и затем вывод заполненных элементов массива в удобной для просмотра на экране (и при распечатке) форме. При выводе на дисплей можно ориентироваться на ширину экрана (80 позиций), при печати – на 60 (вдоль тетрадного листа) или 76 позиций (поперек тетрадного листа или при выводе на стандартный лист).

Обычно для распечатки результатов работы используют вывод в текстовый файл, который затем выводят на принтер, как и текст программы. Чтобы вывод выполнять в файл, необходимо предусмотреть в программе следующие шаги:

- в разделе описаний переменных завести файловую переменную типа text для вывода;
- в выполняемом блоке связать файловую переменную с набором данных и открыть для вывода (используя стандартные процедуры Assign и ReWrite);
- в операторах Write и WriteLn в качестве первого (или

единственного) параметра указывать имя файловой переменной;
- перед концом программы закрыть выводной файл стандартной процедурой Close.

Оформление вывода следует выполнять с использованием формата, размер которого определяется максимальными размерами выводимых значений.

Например, если числа целые и находятся в диапазоне от - 999 до + 999, формат должен быть не менее :5 (с учетом разделяющих пробелов). Если диапазон целых чисел неизвестен, следует рассчитывать на максимум. Для самых длинных целых чисел (от - 32768 до + 32767) он составит :7, и выводить удобно по 10 значений в строке.

Для вещественных чисел, если использовать экспоненциальную форму записи, достаточно оставлять три значащие цифры, что с учетом знака, точки и порядка числа составит :11, например, - 0.836E-02, и выводить имеет смысл по пять чисел в строке.

Конечно, удобнее числа выводить в форме с фиксированной точкой (что можно делать, если порядки чисел известны и они не сильно отличаются от нулевого). Например, по формату :8:2 вывод идет с точностью до сотых.

При выводе в конце каждой строки следует давать команду Writeln для перехода на новую строку. Определить, что пора менять строку, можно по остатку от деления текущего номера элемента на количество значений в строке. Ниже приведен пример вывода на печать в выводной текстовый файл Fout одномерного вещественного массива по "k" значений в строке:

```
Assign(Fout,'UMNIK4.RES'); {связали Fout с Н.Д. UMNIK4.RES }
ReWrite(Fout);          { открыли файл для записи }
Writeln(Fout,' Исходный массив из ',N,' элементов');
for i:=1 to N do
Begin
  Write(Fout,Massiv2[i]:8:2); { печать в текущей строке}
  if i mod k = 0 then Writeln(Fout); {если номер элемента
                кратен "к", переходим на новую строку}
End;
. . .
Close(Fout);   {закрытие файла }
```

3. Обработка массива. Здесь могут встретиться различные ситуации, на которые следует обратить внимание:

а) если в задаче предлагается использовать признак делимости на некоторое число значения переменной (например, требуется что-то делать с каждым элементом массива, который нацело делится на 5), то следует в цикле брать каждый элемент и проверять на равенство нулю остатка от деления значения элемента массива на 5:

```
for i:=1 to N do
begin
  if K[i] mod 5 = 0 then
  { что надо делать, т.к. элемент массива делится на 5}
...
end;
```

б) если же предлагается использовать признак делимости номера элемента, то разумнее сначала вычислить количество выбираемых номеров и организовать цикл по вычисленному количеству, каждый раз определяя, какой элемент массива надо обрабатывать.

Например, если надо что-то делать с каждым пятым элементом массива *K* длиной *N* элементов, то цикл обработки будет иметь вид:

```
M:=N div 5;  { определение числа повторений цикла }
for i:=1 to M do
begin
  j:=i*5; { определение номера обрабатываемого элемента}
  {  обработка K[j]-го элемента массива }
  . . .
end;
```

Если нужно обрабатывать каждый пятый элемент, начиная с третьего, программа будет выглядеть по-другому:

```
j:=3;
M:=(N-(3-1)) div 5;{определение числа повторений цикла}
for i:=1 to M do
begin    {  обработка K[j]-го элемента массива }
  . . .
  j:=j+5; {определение номера следующего обраб.элемента}
end;
```

в) если требуется обрабатывать символьный массив, например, подсчитать, сколько раз встречается символ '®', нужно

уметь, во-первых, присваивать символьным переменным нужные значения, во-вторых, вводить символы в массив с клавиатуры, и, в-третьих, уметь сравнивать символьные элементы массива;

Присваивание значения переменной можно сделать либо перенося это значение из другой переменной (если она содержит в данный момент нужный символ), либо задавая присваивание константы, либо используя функцию Chr(i), преобразования целого числа в символ (с кодом, равным этому числу). Таблица кодов символов приведена в Приложении 2. Примеры присваивания:

```
  Const
     SimA ='A';
     Paragraf = #21;
     Kod = 21;
     SixKod = $15;
  Var S1,S2,S3,S4,S5,S6,S7 : char;
   . . .
  Begin
   . . .
   S1:= SimA;
   S2:=S1;
   S3:=Paragraf;
   S4:=#21;
   S5:=#$15;
   S6:=Chr(21);
   S7:=Chr(15);
```

Пример используемых операторов:

```
. . .
Var Sim : char;
  TextM:array[1..100] of char;{завели символьный массив}
  i,N : shortint;
   . . .
Begin
   . . .
{Если заполнение символьного массива TextM с клавиатуры}
  for i:=1 to N do
  begin
    writeln('Вводи очередной символ');
    readln(TextM[i]);
  end;
```

```
 . . .
{Если заполнение символьного массива TextM из файла}
 assign(fin,'LABA4.DAT');
 reset(fin);
 for i:=1 to N do
  read(fin,TextM[i]);
 . . .
 close(fin); { не забыть в конце закрыть все открытые файлы }
```

4. Вывод результатов. Если в процессе выполнения программы исходный массив изменяется (в нем меняются сами значения элементов, их количество или они переставляются), исправленный массив должен выводиться в конце программы под заголовком 'Измененный массив'.

Пример выполнения задания приведен ниже, данные для задания приведены в табл.16.

Разбор контрольного варианта

Задание

Таблица 16. Данные к заданию 26-го варианта

№ вар.	Задание	Печатать элементы массива по		Тип обрабатываемых данных
		Штук	Формату	
26	Формирование символьного массива длиной не более ста элементов, заполнение его с клавиатуры (вводя по одному произвольному символу пока не встретится символ "."), подсчет и вывод на экран символа, который встретился чаще других, и числа его повторений.	30	:2	Символьный

Для решения этой задачи нужно выделить место на символьный массив из ста элементов, заполнить его (в итеративном цикле – пока не встретится символ '.' или не введется сто

символов), распечатать (по 30 символов в строке) и провести подсчеты частоты вхождения каждого символа. Последняя часть представляет наибольшую сложность, поэтому остановимся на ней подробнее.

Так как нужно найти самый частый символ и количество его повторений, необходимо в программе завести переменные символьного типа для рассматриваемого символа (SimI) и для самого частого символа (SimMax), а также счетчики для рассматриваемого (Ni) и самого частого (Nmax) символов. В качестве SimI будем брать по очереди каждый символ из массива (во внешнем цикле по i – номеру в массиве длиной N элементов) и для него считать, сколько раз этот символ встречается в массиве (внутренний цикл по j – для всех элементов массива). Если после подсчета Ni окажется больше Nmax, значение Ni переносится в Nmax, а SimI – в SimMax.

Очевидно, перед внешним циклом Ni следует обнулить, а в SimMax занести, например, первый элемент символьного массива. Длина массива N подсчитывается один раз при заполнении массива.

Для разбираемого варианта не приводятся таблица идентификаторов и блок-схема алгоритма, хотя при оформлении лабораторной работы их следует включить в отчет.

С учетом выполненного выше разбора задачи, программа может иметь вид:

Текст программы

```
PROGRAM  Simbol_Array;
{  Программа Лабораторной работы N 4
        Вариант N 26.
   П.Г. Петров, ст. гр.990 }
VAR
 N,Ni,Nmax,i,j :integer;
 SimI,SimMax   :char;
 Simbol        :array[1..100] of char;
 Fout          :text;
BEGIN   {Выполняемый блок. Первый раздел – ввод данных }
 N:=0;
 writeln('Вводить по одному не более 100 символов,');
 writeln(' Для завершения – ввести символ "."');
 repeat
```

```
  writeln('Введи очередной символ');
  N := N + 1;
  readln(Simbol[N]);
 until (N = 100) or (Simbol[N] = '.');
 if Simbol[N]= '.' then N := N - 1; { Последний символ, если он –
точка, рассматриваться не будет }
 if N < 1 then
  writeln('Нельзя рассматривать меньше 1 символа')
 else
 begin
  Assign(Fout,'LAB4.RES'); {связали Fout с LAB.DAT}
  ReWrite(Fout);          {открыли файл для записи }
  writeln(Fout,'Исходный массив из ',N,' элементов');
  for i:=1 to N do
  begin
   write(Fout,Simbol[i]:2); { печать в текущей строке}
   if i mod 30 = 0 then writeln(Fout); { если номер
         элемента кратен 30, переходим на новую строку}
  end;
  writeln(Fout);
     { Основной раздел – поиск номера }
  Nmax := 0;  { количество наиболее частого символа }
  SimMax := Simbol[1];
  for i:=1 to N do { внешний цикл перебора символов }
  begin
   SimI := Simbol[i];
   Ni := 0;
   for j:=i to N do { внутренний цикл перебора символов }
    if Simbol[j] = SimI then Ni:=Ni+1;
   if Ni>Nmax then {если этот символ встретился чаще,}
   begin        {запомним его и число его вхождений}
    Nmax := Ni;
    SimMax := SimI;
   end;
  end;

    { Раздел вывода результатов поиска }
 if Nmax = 1 then
  writeln(Fout,'Все символы входят по 1 разу')
```



```
  else
   write(Fout,'Символ "',SimMax,'" встретился ',Nmax,
            ' раз');
     { далее определим, если Nmax кончается на 2,3,4 и не во
втором десятке, изменим окончание, например, "23 раза" }
    if (Nmax mod 10 < 5) and (Nmax mod 10 > 1) and
     (Nmax div 10 <> 1)
    then writeln(Fout,'а');
 end;
 Close(Fout);
END.
```

Результаты работы
(содержимое файла UMNIK.RES)

```
Исходный массив из 97 элементов
В м и р е с у щ е с т в у ю т д в е б е з г р а н и ч н ы е в е щ и – э т о
В с е л е н н а я и ч е л о в е ч е с к а я г л у п о с т ь, х о т я н а с ч е т
В с е л е н н о й я н е у в е р е н
```

Символ "е" встретился 17 раз .

Варианты заданий для выполнения работы приведены в табл.17.

Варианты заданий

Таблица 17. Варианты заданий лабораторной работы № 4

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
1	Формирование массива длиной не более ста элементов, заполнение его с клавиатуры (вводя по одному произвольному символу, пока не встретится символ '!'), исключение из массива всех символов – цифр и пробелов со сдвигом остающихся элементов массива на освободившиеся места.	30	:1	Символьн.
2	Нахождение номера наименьшего элемента в массиве заданной длины среди всех положительных элементов. Длина массива вводится с клавиатуры.	5	:11	Веществ.
3	Вычисление суммы "*K*" слагаемых – элементов массива, начиная с третьего по порядку, и суммируя только элементы с нечетными номерами. Длина массива и количество суммируемых элементов вводится с клавиатуры.	6	:9:2	Веществ.
4	Перестановка максимального и минимального по значению элементов массива. Значение длины массива вводится с клавиатуры.	8	:7	Целые
5	Вычисление суммы всех четных (по значению) элементов массива, расположенных на нечетных по порядку местах. Длина массива вводится с клавиатуры.	10	:6	Целые

Продолжение табл. 17

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
6	Нахождение максимального значения в массиве среди всех четных чисел. Длина массива вводится с клавиатуры.	9	:6	Целые
7	Вычисление суммы элементов массива, начиная с «*K*»-го по порядку и до элемента, равного нулю (если такой элемент встретится; иначе – до конца массива). «*K*», длину массива и значения элементов вводить с клавиатуры.	5	:12	Веществ.
8	Перестановка «i»-го и «j»-го по порядку элементов массива при условии, что они с разными знаками. Если они с одинаковыми знаками, все элементы между ними обнулить. Длина массива и номера переставляемых элементов вводятся с клавиатуры.	10	:6	Целые
9	Вычисление суммы элементов массива, расположенных в конце массива, причем складывать нужно, начиная от конца массива, столько элементов, пока сумма не превысит значения 20.5 (или не будут сложены все элементы). Длина массива вводится с клавиатуры. Печатать сумму и количество сложенных элементов	6	:10:3	Веществ.

Продолжение табл.17

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
10	Нахождение максимального значения в массиве среди всех элементов после первого отрицательного. Длина и значения элементов массива вводятся с клавиатуры.	10	:5	Целые
11	Вычисление суммы элементов массива начиная с первого элемента со значением больше 0.9, и пока сумма по модулю не превысит заданного значения. Длина массива, значения элементов и предельное значение (признак для окончания суммирования) вводятся с клавиатуры.	5	:9:3	Веществ.
12	Перестановка максимального и минимального по коду символов массива длиной N элементов. Значение длины массива и элементы массива вводятся с клавиатуры. Массив печатать до и после перестановки.	30	:2	Символьн.
13	Формирование массива длиной N элементов, заполняя его с клавиатуры (вводя сначала количество символов, и затем по одному произвольному символу, пока не введется указанное количество), подсчет и вывод на экран количества символов из диапазона от 'A' до 'я' (кириллицы).	30	:2	Символьн.

Продолжение табл. 17

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
14	Вычисление среднего значения в одномерном массиве для всех элементов между первым и вторым нулем в массиве (или от первого нулевого значения до конца массива). Длина и значения элементов массива вводятся с клавиатуры.	5	:12	Веществ.
15	Перестановка одномерного массива в обратном порядке. Значение длины массива вводится с клавиатуры. Массив печатать до и после перестановки.	9	:8	Целые
16	Вычисление в одномерном массиве целой части среднего значения всех положительных четных по величине чисел. Длина массива вводится с клавиатуры.	8	:8	Целые
17	Исключение из массива пробелов и запятых со сдвигом остающихся элементов массива на освободившиеся места. Исходный массив длиной N символов(где N не больше 100) вводится из текстового файла.	30	:1	Символьн.
18	Вычисление целого среднеарифметического значения всех отрицательных элементов массива (содержащего и положительные значения), расположенных начиная с «*K*»-го по порядку элемента. Длина массива и значение «*K*» вводятся с клавиатуры.	10	:6	Целые

Продолжение табл. 17

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
19	Определение длины самой длинной последовательности из расположенных подряд в одномерном логическом массиве значений TRUE и вывод найденной длины на экран. Длина массива и значения TRUE и FALSE вводятся с клавиатуры в форме *F* (для FALSE) и *T* (TRUE).	8	:7	Логич.
20	Вычисление среднеарифметического всех положительных элементов массива (содержащего и отрицательные значения), длина которого вводится с клавиатуры.	6	:10:2	Веществ.
21	Поиск места (номера элемента) в массиве, где первый раз подряд встречаются два четных числа. Длина массива вводится с клавиатуры.	10	:6	Целые
22	Формирование логического массива длиной N элементов, заполнение его с клавиатуры (вводя 1 – вместо TRUE и 0 – вместо FALSE), подсчет и вывод на экран количества значений TRUE и FALSE и сообщение, чего было больше.	10	:6	Логич.

Окончание табл. 17

№ вар.	Задание	Печатать элементы массива по		Тип обрабат. данных
		штук	формату	
23	Поиск номеров двух последних расположенных подряд отрицательных элементов в массиве. Длина массива вводится с клавиатуры.	6	:10	Веществ.
24	Формирование логического массива, заполняя его с клавиатуры (вводя вместо TRUE четные числа, а вместо FALSE – нечетные, и заканчивая, когда встретится число 0); подсчет и вывод на экран количества значений TRUE и FALSE и сообщение, каких значений было больше.	12	:5	Логич.
25	Нахождение номера максимального значения в массиве после первого отрицательного и не далее второго отрицательного. Длина массива вводится с клавиатуры.	5	:10:3	Веществ.
26	Формирование символьного массива длиной не более ста элементов, заполнение его с клавиатуры (вводя по одному произвольному символу пока не встретится символ "."), подсчет и вывод на экран символа, который встретился чаще других и число его повторений	30	:2	Символьн.

Библиографический список

1. Острейковский В.А. Информатика: учебник для вузов.– М.: Высш.шк., 2000.
2. Вирт Н. Алгоритмы и структура данных. – М.: Мир,1989.
3. Лабораторый практикум по информатике: учебное пособие для вузов /В.С.Микшина, Г.А.Еремеева, Н.Б.Назина и др.; под ред. В.А.Острейковского. – 2-е изд., стер. – М.: Высш. шк., 2006.
4. Фаронов В.В. Турбо Паскаль 7.0. Начальный курс: учебное пособие. – М.: Нолидж, 1997.

ПРИЛОЖЕНИЕ 1
Правила оформления отчета по лабораторной работе

Отчет должен быть оформлен на стандартных белых листах формата А4 (210х297 мм) на одной стороне листа. Отчет должен содержать титульный лист и листы с выполненной работой.

Титульный лист оформляется в соответствии с правилами оформления расчетно-графических работ. Пример титульного листа приведен на следующей странице.

Остальные листы предназначены для печати результатов выполнения лабораторной работы и в каждой работе должны содержать те пункты, которые требуются для ее выполнения.

Федеральное агентство по образованию
Санкт-Петербургский государственный технологический университет
растительных полимеров

Кафедра прикладной математики и информатики

Лабораторная работа № __
"Название работы"
(Вариант №__)

Выполнил (а) студент(ка) гр.__ ф.и.о. студента

Проверил доцент кафедры ПМиИ Антонюк П.Е.

Санкт-Петербург
20..

ПРИЛОЖЕНИЕ 2

Таблица ASCII-кодов (с альтериативной кодировкой)

Первая половина таблицы – стандартный набор кодов символов

№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ
0	00	(null)	32	20		64	40	@	96	60	`
1	01	☺	33	21	!	65	41	A	97	61	A
2	02	☻	34	22	"	66	42	B	98	62	B
3	03	♥	35	23	#	67	43	C	99	63	C
4	04	♦	36	24	$	68	44	D	100	64	d
5	05	♣	37	25	%	69	45	E	101	65	e
6	06	♠	38	26	&	70	46	F	102	66	f
7	07	•	39	27	'	71	47	G	103	67	g
8	08	◘	40	28	(	72	48	H	104	68	h
9	09	○	41	29	)	73	49	I	105	69	i
10	0A	■	42	2A	*	74	4A	J	106	6A	j
11	0B	♂	43	2B	+	75	4B	K	107	6B	k
12	0C	♀	44	2C	,	76	4C	L	108	6C	l
13	0D	♪	45	2D	-	77	4D	M	109	6D	m
14	0E	♫	46	2E	.	78	4E	N	110	6E	n
15	0F	☼	47	2F	/	79	4F	O	111	6F	o
16	10	►	48	30	0	80	50	P	112	70	p
17	11	◄	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	‼	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	↨	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[	123	7B	{
28	1C	└	60	3C	<	92	5C	\	124	7C	|
29	1D	↔	61	3D	=	93	5D	]	125	7D	}|
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	_	127	7F	Δ

Вторая половина таблицы – альтернативный набор кодов символов .

№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ	№ п/п	Код 16-й	Символ
128	80	А	160	A0	а	192	C0	└	224	E0	р
129	81	Б	161	A1	б	193	C1	┴	225	E1	с
130	82	В	162	A2	в	194	C2	┬	226	E2	т
131	83	Г	163	A3	г	195	C3	├	227	E3	у
132	84	Д	164	A4	д	196	C4	─	228	E4	ф
133	85	Е	165	A5	е	197	C5	┼	229	E5	х
134	86	Ж	166	A6	ж	198	C6	╞	230	E6	ц
135	87	З	167	A7	з	199	C7	╟	231	E7	ч
136	88	И	168	A8	и	200	C8	╚	232	E8	ш
137	89	Й	169	A9	й	201	C9	╔	233	E9	щ
138	8A	К	170	AA	к	202	CA	╩	234	EA	ъ
139	8B	Л	171	AB	л	203	CB	╦	235	EB	ы
140	8C	М	172	AC	м	204	CC	╠	236	EC	ь
141	8D	Н	173	AD	н	205	CD	═	237	ED	э
142	8E	О	174	AE	о	206	CE	╬	238	EE	ю
143	8F	П	175	AF	п	207	CF	╧	239	EF	я
144	90	Р	176	B0	▒	208	D0	╨	240	F0	Ё
145	91	С	177	B1	▓	209	D1	╤	241	F1	ё
146	92	Т	178	B2	▓	210	D2	╥	242	F2	Є
147	93	У	179	B3	¦	211	D3	╙	243	F3	є
148	94	Ф	180	B4	┤	212	D4	╘	244	F4	Ї
149	95	Х	181	B5	╡	213	D5	╒	245	F5	ї
150	96	Ц	182	B6	╢	214	D6	╓	246	F6	Ў
151	97	Ч	183	B7	╖	215	D7	╫	247	F7	ў
152	98	Ш	184	B8	╕	215	D8	╪	248	F8	°
153	99	Щ	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ъ	186	BA	¬	218	DA	┌	250	FA	·
155	9B	Ы	187	BB	╗	219	DB	▉	251	FB	√
156	9C	Ь	188	BC	╝	220	DC	▄	252	FC	№
157	9D	Э	189	BD	╜	221	DD	▌	253	FD	¤
158	9E	Ю	190	BE	╛	222	DE	▐	254	FE	■
159	9F	Я	191	BF	┐	223	DF	▀	255	FF	

СОДЕРЖАНИЕ

ВВЕДЕНИЕ....3
1. ПОНЯТИЕ АЛГОРИТМА....4
2. СПОСОБЫ ОПИСАНИЯ АЛГОРИТМА....7
3. ОСНОВНЫЕ ВИДЫ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ 10
4. ЛАБОРАТОРНЫЕ РАБОТЫ....13
Общая схема выполнения лабораторной работы ....–
Лабораторная работа № 1. Алгоритмы линейной структуры 15
Лабораторная работа № 2. Программирование алгоритмов с ветвлениями...26
Лабораторная работа № 3. Работа с последовательностями чисел 37
Лабораторная работа № 4. Работа с одномерными массивами (подсчет, поиск элементов, перестановки в массиве)....51
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 68
ПРИЛОЖЕНИЕ 1. ПРАВИЛА ОФОРМЛЕНИЯ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ 68
ПРИЛОЖЕНИЕ 2. ТАБЛИЦА ASCII –КОДОВ (С АЛЬТЕРНАТИВНОЙ КОДИРОВКОЙ) 70

Редактор и корректор Н.П.Новикова

Техн.редактор Л.Я.Титова

Темплан 2009, поз. 25

Подп. к печати 16.02.2009. Формат 60x84/16. Бумага тип. №1. Печать офсетная. Усл.печ.л. 4,5. Уч.-изд.л. 4,5.Тираж 200 экз. Заказ № 2023

Ризограф ГОУВПО Санкт-Петербургского государственного технологического университета растительных полимеров, 198095,Санкт-Петербург, ул.Ивана Черных,4.