ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО
ОБРАЗОВАНИЯ ИРКУТСКОЙ ОБЛАСТИ
«БРАТСКИЙ ПРОМЫШЛЕННО – ГУМАНИТАРНЫЙ ТЕХНИКУМ»

СБОРНИК МЕТОДИЧЕСКИХ УКАЗАНИЙ

**по выполнению практических работ по
учебной дисциплине**

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Структурное программирование

230401 Информационные системы (в строительстве)

«профессиональный цикл»

Братск, 2013 г.

Сборник методических указаний по выполнению практических работ составлен в соответствии с требованиями к результатам освоения УД «Основы алгоритмизации и программирования», изложенными в Федеральном государственном образовательном стандарте среднего профессионального образования по специальности **230401 «Информационные системы (в строительстве)».**

Организация: Государственное бюджетное образовательное учреждение среднего профессионального образования Иркутской области «Братский промышленно-гуманитарный техникум»

Автор-составитель: Янина Е.А., преподаватель ГБОУ БПГТТ

Сборник описаний практических работ одобрен на заседании цикловой комиссии информационно-гуманитарных дисциплин

Протокол №___ от _____20____г.

Председатель ЦК _____Н.А. Орлова

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

ПРАКТИЧЕСКАЯ РАБОТА 1. ТЕМА: СОСТАВЛЕНИЕ БЛОК-СХЕМ АЛГОРИТМОВ....5

ПРАКТИЧЕСКАЯ РАБОТА 2ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ 9

ПРАКТИЧЕСКАЯ РАБОТА 3. ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ.....12

ПРАКТИЧЕСКАЯ РАБОТА 4.ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ УСЛОЖНЕННОЙ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ.....18

ПРАКТИЧЕСКАЯ РАБОТА 5. ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ 20

ПРАКТИЧЕСКАЯ РАБОТА 6. ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ УСЛОЖНЕННОЙ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

ПРАКТИЧЕСКАЯ РАБОТА 7. ТЕМА: ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ.....30

ПРАКТИЧЕСКАЯ РАБОТА 8. ТЕМА: ОБРАБОТКА МНОГОМЕРНЫХ МАССИВОВ....35

ПРАКТИЧЕСКАЯ РАБОТА 9. ТЕМА: ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ФУНКЦИЙ И ПРОЦЕДУР ДЛЯ РАБОТЫ СО СТРОКАМИ 42

ПРАКТИЧЕСКАЯ РАБОТА 10. ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ СО СТРУКТУРИРОВАННЫМ ТИПОМ ДАННЫХ «МНОЖЕСТВО» 46

ПРАКТИЧЕСКАЯ РАБОТА 11. ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕ ПРОЦЕДУР. ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ.....48

ПРАКТИЧЕСКАЯ РАБОТА 12. ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ТЕКСТОВЫХ ФАЙЛОВ

ПРАКТИЧЕСКАЯ РАБОТА 13. ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕ ТИПИЗИРОВАННЫХ И НЕТИПИЗИРОВАННЫХ ФАЙЛОВ.....59

ПРАКТИЧЕСКАЯ РАБОТА 14. ТЕМА: ПРОГРАММИРОВАНИЕ МОДУЛЯ.....62

ВВЕДЕНИЕ

Основное содержание методических указаний по выполнению практических работ ориентировано на обучение структурной методике программирования. Поэтому первый практические работы предназначены для освоения программирования основных алгоритмических конструкций: линейной, ветвлению, циклам, а также способам составления подпрограмм. Последующие практические работы посвящены работе с разными видами структур данных: массивы, записи, множества, файлы.

Методические указания предназначены для изучения программирования на языке Паскаль.

Методические указания по учебной дисциплине «Основы алгоритмизации и программирования» для выполнения практических работ созданы Вам в помощь для работы на занятиях, подготовки к практическим занятиям.

Приступая к выполнению практической работы, Вы должны внимательно прочитать цель, краткие теоретические и учебно-методические материалы по теме лабораторно-практической работы, выполнить самостоятельные задания и ответить на вопросы для закрепления теоретического материала.

Все задания к лабораторной или практической работе Вы должны выполнять в соответствии с инструкцией, анализировать полученные в ходе занятия результаты.

В результате освоения учебной дисциплины «Основы алгоритмизации и программирования» Вы приобретете умения:

- использовать языки программирования
- строить логически правильные и эффективные программы;

В результате освоения учебной дисциплины Вы будете знать:

- общие принципы построения алгоритмов, основные алгоритмические конструкции;
- понятие системы программирования;
- основные элементы процедурного языка программирования, структура программы, операторы и операции, управляющие структуры, структуры данных, файлы, кассы памяти;
- подпрограммы, составление библиотек программ;
- объектно-ориентированная модель программирования, понятие классов и объектов, их свойств и методов

Наличие положительной оценки по практическим работам необходимо для получения допуска к экзамену, поэтому в случае отсутствия на уроке по любой причине или получения неудовлетворительной оценки за практическую работу Вы должны найти время для ее выполнения или пересдачи.

Внимание! Если в процессе подготовки к лабораторно-практическим работам или при решении задач у Вас возникают вопросы, разрешить которые самостоятельно не удается, необходимо обратиться к преподавателю для получения разъяснений или указаний в дни проведения консультаций.

Время проведения дополнительных занятий (консультаций) можно узнать у преподавателя или посмотреть на стенде в учебном корпусе.

Желаем Вам успехов!!!

ПРАКТИЧЕСКАЯ РАБОТА 1

ТЕМА: СОСТАВЛЕНИЕ БЛОК-СХЕМ АЛГОРИТМОВ

Цель работы: научиться составлять алгоритмы графическим способом (блок-схем)

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Этапы решения задачи на ЭВМ. Работа по решению любой задачи с использованием компьютера включает в себя шесть этапов
1) постановка задачи
2) формализация задачи
3) построение алгоритма
4) составление программы на языке программирования
5) отладка и тестирование программы
6) проведение расчетов и анализ полученных результатов

Часто эту последовательность называют технологической цепочкой решения задачи на ЭВМ.

На этапе постановки задачи следует четко определить, что дано и что требуется найти. Важно описать полный набор исходных данных, необходимых для решения задачи На этапе формализации чаще всего задача переводится на язык математических формул, уравнений и отношений. Если
решение задачи требует математического описания какого-то реального объекта, явления или процесса, то ее формализация равносильна получению соответствующей математической модели

Третий этап — это построение алгоритма. Опытные программисты часто сразу пишут программы на определенном языке, не прибегая к каким-либо специальным средствам описания алгоритмов (блок-схемам, псевдокодам), однако в учебных целях полезно сначала использовать эти средства, а затем переводить полученный алгоритм на язык программирования

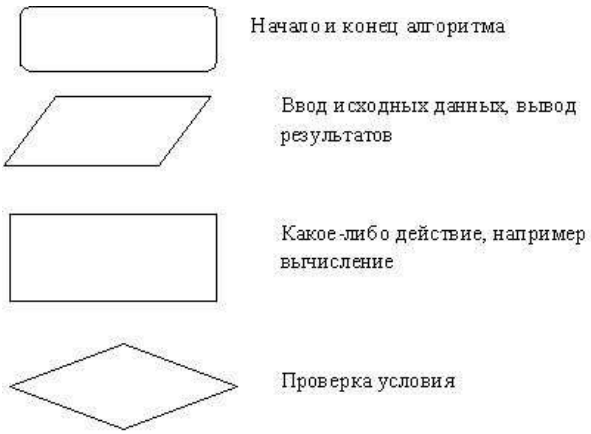
Алгоритм — это последовательность команд управления каким-либо исполнителем. В школьном курсе информатики с понятием алгоритма и
методами построения алгоритмов ученики знакомятся на примерах учебных исполнителей: Робота, Черепахи, Чертежника и др. Эти исполнители ничего не вычисляют. Они создают рисунки на экране, перемещаются в лабиринтах, перетаскивают предметы с места на место.

Данные и величины. Совокупность величин, с которыми работает компьютер, принято называть данными. По отношению к программе различают исходные, окончательные (результаты) и промежуточные данные, которые получают в процессе вычислений.

В каждом языке программирования существует своя концепция и своя система типов данных. Однако в любой язык входит минимально необходимый набор основных типов данных: целые вещественные, логические и символьные. С типом величины связаны три ее свойства: множество допустимых значений, множество допустимых операций, форма внутреннего представления.

Блок-схема — графическое представление алгоритма. Она состоит из функциональных блоков, которые выполняют различные назначения (ввод/вывод, начало/конец, вызов функции и т.д.).

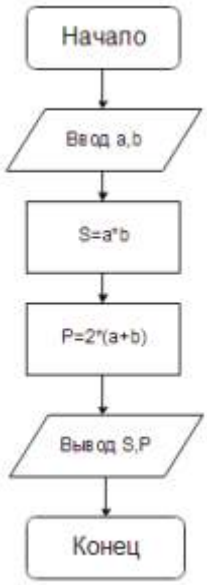
Существует несколько основных видов блоков, которые нетрудно запомнить:



Пример №1: Рассчитать площадь и периметр прямоугольника по двум известным сторонам.
Данная задача не должна представлять особой трудности, так как построена она на хорошо известных всем нам формулах расчета площади и периметра прямоугольника, поэтому зацикливаться на выведении этих формул мы не будем.

Составим алгоритм решения подобных задач:
1) Прочитать задачу.
2) Выписать известные и неизвестные нам переменные в «дано». (В задаче №1 к известным переменным относятся стороны: a,b ;к неизвестным — площадь S и периметр P)
3) Вспомнить либо составить необходимые формулы. (У нас: S=a*b; P=2*(a+b))
4) Составить блок-схему.
5) Записать решение на языке программирования Pascal.
Запишем условие в более кратком виде.
Дано: a,b
Найти: S,P

Блок-схема:



Словесное описание алгоритма:
Структура программы, решающей данную задачу, тоже проста:
1) Описание переменных;
2) Ввод значений сторон прямоугольника;

3) Расчет площади прямоугольника;
4) Расчет периметра прямоугольника;
5) Вывод значений площади и периметра;
6) Конец.

ЗАДАНИЕ

Составить словесно-формульный алгоритм и блок-схему для следующих задач:

1. Вычислить периметр и площадь прямоугольного треугольника по длинам двух катетов: a и b;
2. Вычислить длину окружности и площадь круга с заданным радиусом R/
3. Вычислить расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2)

ПРАКТИЧЕСКАЯ РАБОТА 2

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ

Цель работы: научиться составлять программы линейной структуры, реализовывать в программе оператор присваивания, процедуры ввода/вывода; строить блок-схемы линейной конструкции.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ ДАННЫЕ

Приступая к решению задач этого раздела, следует вспомнить, что:

- программы с линейной структурой являются простейшими и используются, как правило, для реализации обычных вычислений по формулам;
- в программах с линейной структурой инструкции выполняются последовательно, одна за другой;
- алгоритм программы с линейной структурой может быть представлен следующим образом (Рис. 1):

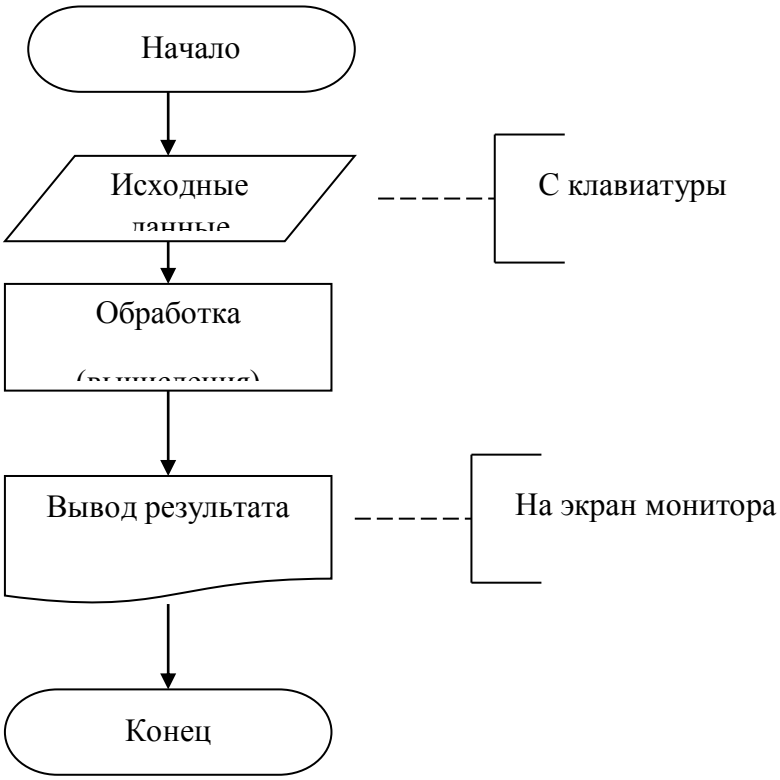


Рис. 1. Блок-схема линейной конструкции

Математические стандартные функции Турбо Паскаля

Обращение	Тип аргумента	Тип результата	Функция
Pi	—	*R*	Число π = 3.1415926536E+00
Abs(x)	*I*, *R*	*I*, *R*	Модуль аргумента *x*
Arctan(x)	*I*, *R*	*R*	Арктангенс *x* (радианы)
Cos(x)	*I*, *R*	*R*	Косинус *x* (*x* в радианах)
Exp(x)	*I*, *R*	*R*	e^x — экспонента
Frac(x)	*I*, *R*	*R*	Дробная часть *x*
Int(x)	*I*, *R*	*R*	Целая часть *x*
Ln(x)	*I*, *R*	*R*	Натуральный логарифм *x*
Random		*R*	Псевдослучайное число в интервале [0, 1)
Random(x)	*I*	*I*	Псевдослучайное число в интервале [0, *x*)
Round(x)	*R*	*I*	Округление до ближайшего целого
Sin(x)	*I*, *R*	*R*	Синус *x* (*x* в радианах)
Sqr(x)	*I*, *R*	*I*, *R*	Квадрат *x*
Sqrt(x)	*I*, *R*	*R*	Корень квадратный из *x*
Trunc(x)	*R*	*I*	Ближайшее целое, не превышающее *x* по модулю

Где I — Integer; R — Real

ЗАДАНИЕ

1. Написать программу вычисления объема цилиндра. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема цилиндра
Введите исходные данные:
Радиус основания (см) —> 5
Высота цилиндра (см) —> 10
Объем цилиндра 1570.80 куб. см.

Для завершения работы программы нажмите <Enter>.

2. Написать программу вычисления объема цилиндра. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление объема цилиндра
Введите исходные данные:
Радиус основания (см) —> 5
Высота цилиндра (см) —> 10
Объем цилиндра 1570.80 куб. см.

Для завершения работы программы нажмите <Enter>.

3. Написать программу вычисления двух выражений:

$$z_1 = \frac{\sqrt{2b + 2\sqrt{b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$$

$$z_2 = \sqrt{\frac{a+b}{a-3}}$$

4. Для каждой программы составить блок-схему

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы данные вы знаете?
2. Что такое алгоритм?
3. Приведите пример алгоритма из реальной жизни
4. Какими свойствами обладает алгоритм?
5. Что такое линейная конструкция?
6. Какие операторы используются для реализации линейной конструкции в программе?
7. Назовите процедуры ввода/вывода данных
8. Что такое формат вывода данных?
9. Перечислите основные разделы программы

ПРАКТИЧЕСКАЯ РАБОТА 3

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы: знать о разветвляющейся конструкции алгоритма; уметь реализовывать алгоритмическую конструкцию ветвление с помощью условного оператора и оператора выбора в программе, написанной на языке Паскаль.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ ДАННЫЕ

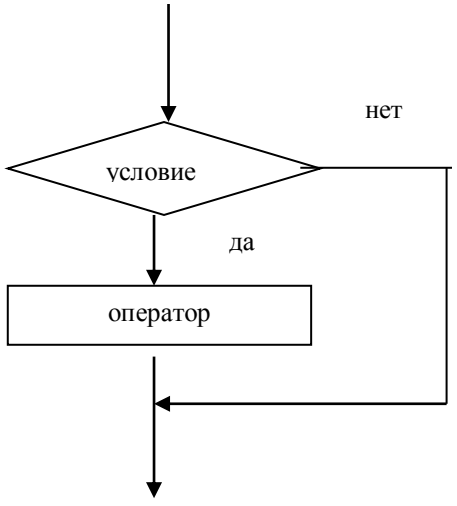
Ветвление — алгоритм, в котором предусмотрены разветвления, указанные в последовательности действий на два направления в зависимости от итогов проверки заданного условия**.** То есть такой алгоритм, обязательно содержит условие и в зависимости от результата выполнения условия происходит выбор действия.

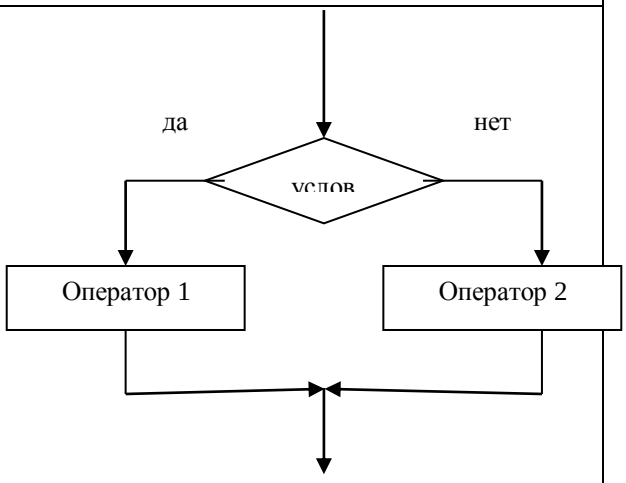
Например: Если день рабочий, то идем в лицей, иначе будем отдыхать. Таких примеров можем привести много из обычной жизни и наук. К примеру, физика: **Если** удар упругий, **то** масса тела сохраняется, **иначе** масса изменяется.

Алгоритмическая конструкция ветвление программируется с помощью **условного оператора If**, который может быть представлен двумя вариантами (Таблица 1).

Условный оператор If

Таблица 1

Конструкция	**Графическое представление блок - схема)**
1 Вариант — неполное ветвление	
If <условие> **Then** <оператор> Неполное ветвление — в зависимости от результата проверки условия либо выполняются действия одной ветви «да» (оператор), либо эти действия не выполняются.	

2 Вариант — полное ветвление	
If <условие> **Then** <оператор 1> **Else** <оператор 2> Полное ветвление — в зависимости от результата проверки условия выполняются только оператор 1 ветви «да» или только оператор 2 ветви «нет»	

Условие — это логическое выражение, которое может принимать одно из двух значений: true (истина — условие выполняется) и false (ложь — условие не выполняется).

В условии используются операции отношения (=, <> , >,<, >=, <=) и логические операции (and (И), or (ИЛИ), xor (исключающее ИЛИ), not (отрицание)). Если требуется проверить несколько условий, их объединяют с помощью логических операций.

Примеры логических выражений:

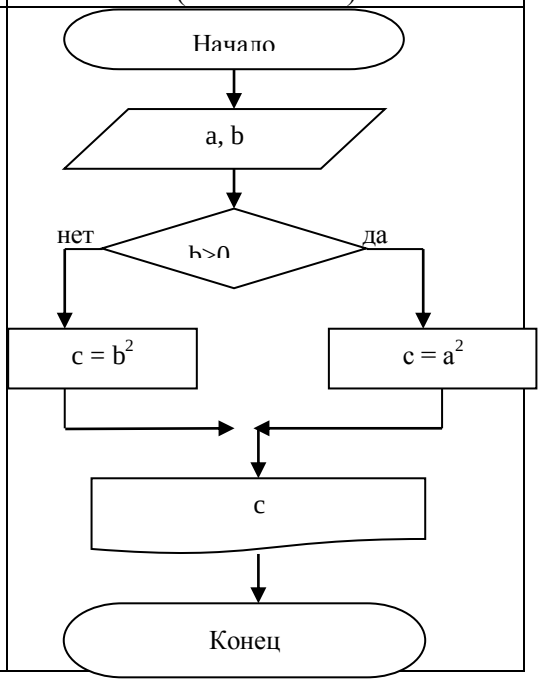
A<2

(x<>0) and (y<>0)

Если между служебными словами стоят несколько операторов, то они заключаются в операторные скобки Begin…End

Рассмотрим пример:

Даны 2 вещественных числа. Если числа положительные, то возвести в квадрат первое число, иначе возвести в квадрат второе число.

Листинг программы	Графическое представление (блок -схема)
Program Primer_1; Uses crt; {Обозначим 1- ое число через переменную a, 2-ое через переменную b, результат — c} Var a, b, c: real; Begin Clrscr; Writeln(‘Введите первое число ’); Readln(a); Writeln(‘Введите второе число ’); Readln(b); **If** (a>0) and (b>0) **Then** c:=sqr(a) **Else** c:=sqr(b); Writeln(‘Результат = ’, c:5:2); End.	

ЗАДАНИЕ

Правила пунктуации при записи операторов

1. Точка с запятой не ставится в разделах описаний после зарезервированных слов uses, label, type, const, var и ставится после завершения каждого описания
2. Точка с запятой не ставится после begin и перед end, так как эти слова являются операторными скобками, а не операторами;
3. Точка с запятой является разграничителем операторов, ее отсутствие между операторами вызывает ошибку компиляции;
4. В операторах цикла точка с запятой не ставится после служебных слов while, repeat, do, until;
5. В условных оператора точка с запятой не ставится после then и перед else

ЗАДАНИЕ

Выполните задание по варианту, назначенному преподавателем.

Вариант 1

Задание 1
Даны три действительные числа. Возвести в квадрат те из них, значения которых положительны, и в четвертую степень — отрицательные.

Задание 2
Написать программу, которая вычисляет частное от деления двух чисел. Программа должна проверять правильность введенных пользователем данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление частного.
Введите в одной строке делимое и делитель,затем нажмите <Enter>
*-> **12 0***
Вы ошиблись. Делитель не должен быть равен нулю.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 2

Задание 1
Даны два действительные числа. Если числа положительны найти их сумму, если отрицательны — произведение

Задание 2
Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные

пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки и нажмите <Enter>
*-> **1200***
Вам предоставляется скидка 10%
Сумма покупки с учетом скидки: 1080.00 руб.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 3

Задание 1
Даны действительные числа x и y, не равные друг другу. Меньшее из этих чисел заменить половиной их суммы, а большее — их удвоенным произведением

Задание 2
Написать программу проверки знания даты основания Санкт-Петербурга. В случае неверного ответа пользователя программа должна выводить правильный ответ. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

В каком году был основан Санкт-Петербург?
Введите число и нажмите <Enter>
*-> **1705***
Вы ошиблись, Санкт-Петербург был основан в 1703 году.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 4

Задание 1
Данные два вещественных числа. Если первое число больше второго, то возвести его в третью степень, если равно второму — прибавить к нему второе число

Задание 2
Написать программу определения стоимости разговора по телефону с учетом скидки 20%, предоставляемой по воскресеньям. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости разговора по телефону.
Введите исходные данные:
*Длительность разговора (целое количество минут) —> **3***
*День недели (1 - понедельник, ... 7 — воскресенье) —> **6***
Предоставляется скидка 20%.
Стоимость разговора: 5.52 руб.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 5

Задание 1

Даны три действительные числа. Если первое число больше второго, умножить данное число на 5, если первое число больше третьего — разделить на два

Задание2
Написать программу — модель анализа пожарного датчика в помещении, которая выводит сообщение «Пожароопасная ситуация», если температура в комнате превысила 60^0C

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 6

Задание 1
Даны действительные числа a, b, c. Удвоить эти числа, если $a \geq b \geq c$, иначе оставить без изменения.

Задание 2
Написать программу, которая анализирует данные о возрасте и относит человека к одной из четырех групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 7

Задание 1
Даны три числа a, b, c. Определить какое из них равно d. Если ни одно не равно d, то найти сумму чисел a, b, c.

Задание 2
Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% — если сумма больше 1000 руб. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки и нажмите <Enter>
-> 640
Вам предоставляется скидка 3%
Сумма покупки с учетом скидки: 620.80 руб.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 8

Задание 1

Даны три действительные числа. Найти минимальное и максимальное число.

Задание 2
Написать программу проверки знания истории архитектуры. Программа должна вывести вопрос и три варианта ответа. Пользователь должен выбрать правильный ответ и ввести его номер. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Архитектор Исаакиевского собора:
1. Доменико Трезини
2. Огюст Монферран
3. Карл Росси
Введите номер правильного ответа и нажмите <Enter>
*-> **3***
Вы ошиблись.
Архитектор Исаакиевского собора — Огюст Монферран.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 9

Задание 1
Даны три действительные числа. Если все числа положительны, найти среднее арифметическое, иначе произведение.

Задание 2
Написать программу проверки знания истории архитектуры. Программа должна вывести вопрос и три варианта ответа, а пользователь — выбрать правильный ответ и ввести его номер. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Невский проспект получил свое название:
1. По имени реки, на берегах которой расположен Санкт-Петербург
2. По имени близко расположенного монастыря Александро-Невской лавры
3. В память о знаменитом полководце Александре Невском
Введите номер правильного ответа и нажмите <Enter>
*-> **3***
Вы ошиблись.
Правильный ответ: 2.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

Вариант 10

Задание 1
Даны два вещественных числа, если числа не равны нулю, возвести из в третью степень, иначе во вторую степень.

Задание 2
Написать программу, которая сравнивает два числа, введенных с клавиатуры. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Ниже представлен рекомендуемый вид экрана во время работы программы.
Введите в одной строке два целых числа
*-> **34 67***
34 меньше 67.

Задание 3
Оформить отчет. Отчет должен содержать коды программ и блок-схемы

ПРАКТИЧЕСКАЯ РАБОТА 4

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ УСЛОЖНЕННОЙ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы: научиться составлять программы с использованием вложенного условия

Оборудование: ПК, ИСР Pascal ABC

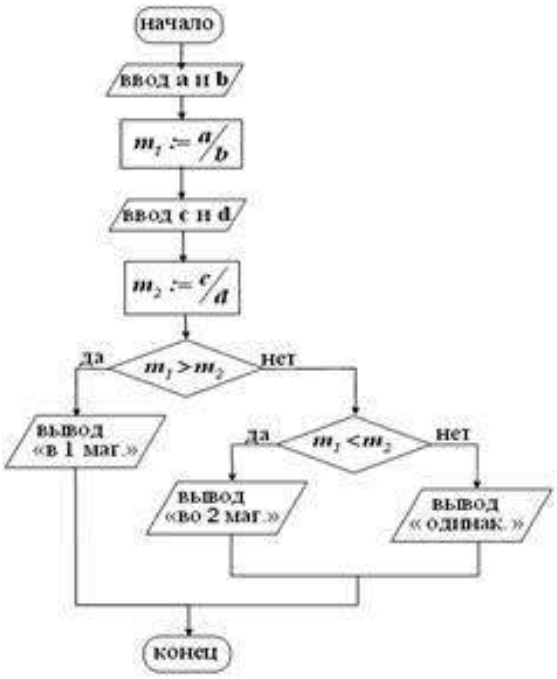
ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Любая ветвь может не быть линейным участком программы, а сама содержать ветвление. Такое ветвление называется вложенным (или множественным) ветвлением.

Чаще вторично разветвляется ветка «нет». В качестве примера разберём простую задачу:

В первом магазине хозяйка приобрела **a** кг. огурцов. Их оказалось **b** штук. Во втором магазине на **c** кг. получилось **d** штук. В каком магазине огурцы крупнее?

Находим массу одного огурца в каждом магазине и сравниваем их. Блок-схема данной задачи представлена ниже.



Рассмотрим пример 2

Написать программу, которая вычисляет оптимальный вес пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправиться или похудеть. Оптимальный вес вычисляется по формуле: **рост (в сантиметрах) — 100.**

Рекомендуемый вид экрана во время работы программы приведен ниже:

Введите в одной строке через пробел
рост (см) и вес (кг) затем нажмите <Enter>
*-> **170 68***
Вам надо поправиться на 2.00 кг.

Листинг программы

```
Program Ves;
var
w: real;        {вес}
h: real;        {рост}
opt: real;      {оптимальный вес}
d: real;        {отклонение от оптимального веса}
begin
        writeln('Введите в одной строке через пробел');
        writeln ('рост (см) и вес (кг), затем нажмите <Enter>');
        write('->');
        readln(h,w);
        opt:=h-100;
        if w=opt then writeln('Ваш вес оптимален!') else
        if w<opt then begin d:=opt-w; writeln('Вам надо поправиться на ',d:5:2,' кг.'); end else
        begin d:=w-opt; writeln('Вам надо похудеть на1, d:5:2,' кг'); end;

end.
```

ЗАДАНИЕ

1. Одна коробка с яйцами содержит 10 ячеек по 30 яиц в каждой. Поместятся ли **a** яиц в **b** коробок (уже имеющих пустые ячейки)? Если не поместятся, сообщить, сколько ещё требуется ячеек и коробок. Если останутся лишние коробки, сообщить, сколько осталось.
2. Определить, являются ли три введённых числа длинами сторон прямоугольного треугольника.
3. В котёл с 20 л воды всыпали **m** граммов соли. Норма для супа составляет от 10 до 12 г/литр. Определить, нормально ли посолена вода. Если недосолена, сообщить, сколько граммов соли нужно добавить до нормы. Если пересолена, — сколько литров воды нужно долить до нормальной концентрации.
4. После промывки шерсть сушат. Нормальная плотность шерсти, соответствующей требуемой влажности, составляет около 280 кг/м^3. На текстильный завод поступило **m** тонн шерсти, объём которой составляет **V** м^3. Определить соответствие сырья требуемой влажности.
5. Определить, лежит ли точка (**x, y**) внутри кольца с центром в начале координат, внутренним радиусом $\mathbf{r_1}$ и внешним радиусом $\mathbf{r_2}$
6. Врач прописал больному первого лекарства всего **a** таблеток, по **b** таблеток в день и второго лекарства **c** таблеток по **d** таблеток в день. На следующий день после того, как все лекарства будет приняты, больной должен прийти на приём. Через сколько дней больной попадёт на приём к врачу?

ПРАКТИЧЕСКАЯ РАБОТА 5

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель работы: научиться использовать операторы циклов при составлении программ на языке Паскаль; составлять блок-схему циклической структуры

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ ДАННЫЕ

Операторы цикла используются для многократного повторения аналогичных вычислений.

Для организации цикла в Паскале имеются три различных оператора.

1. Регулярный оператор For

For <параметр цикла>:=<начальное значение> **to** <конечное значение> **do** S**;**
S- простой или составной оператор.

инструкция for используется для организации циклов с фиксированным, определяемым во время разработки программы, числом повторений;
количество повторений цикла определяется начальным и конечным значениями переменной-счетчика;
переменная-счетчик должна быть целого типа (integer).

При каждом прохождении цикла < параметр цикла >, начиная с <начального значения>, увеличивается на единицу. Цикл выполняется, пока <параметр цикла> не станет больше <конечного значения>.

Другой вариант записи оператора For:
For <параметр цикла >:=< начальное значение> **downto** <конечное значение> **do** S**;**

В этом случае при каждом прохождении цикла <параметр цикла> уменьшается на единицу от <начального значения> до <конечного значения>.

2. Оператор цикла While с проверкой предусловия:

While <условие> **do** S; {Пока выполняется условие, делать}

Цикл выполняется, пока условие истинно (true).

3. Оператор цикла **Repeat** с проверкой постусловия:

Repeat S **until** <условие>; {Выполнять до тех пор, пока не будет выполнено условие}

Цикл выполняется, пока условие ложно (false).

Пример

Постановка задачи. Найти сумму 5 целых чисел от 1 до 5. Написать программы для определения суммы с помощью трех рассмотренных операторов цикла.

Структограмма и программа приведены в таблице 1

Таблица 1. Операторы цикла

Цикл **For** …	**While**…	**Repeat**…	
3.1. Описание S,i; S:=0; For i:=1 to 5 do S:=S+i; Вывод S	3.2 Описание S,i; S:=0; :=1; While :<=5 do S:=S+i; I:=i+1; Вывод S	3.3 Описание S,i; S:=0; i:=1; S=S+i; I:=i+1; Until i>=6; Вывод S	**Структограммы**

4.1:
Program P2;
Var i,S:integer;
Begin
S:=0;
For i:=1 to 5 do
S:=S+i;
Writeln('S=',S:5);
End.
	4.2:
	Program P2;
	Var i,S:byte;
	Begin
	S:=0; i:=1;
	While i<=5 do
	Begin
	S:=S+i;
	I:=i+1;
	End;
	Writeln('S=',S);
	End.
		4.3:
		Program P3;
		Var i,S:integer;
		Begin
		S:=0; i:=1;
		Repeat
		S:=S+i;
		I:=i+1;
		Until i>=6;
		Writeln(S);
		End.
			Текст программ

ЗАДАНИЕ

Задание 1. Составьте программы, используя регулярный оператор цикла согласно своему варианту. Номер варианта соответствует номеру вашего рабочего ПК.

Вариант 1

1. Написать программу, которая выводит таблицу квадратов первых десяти чисел. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов

Число	**Квадрат**
1	**1**
2	**4**
3	**9**
4	**16**
5	**25**
6	**36**
7	**49**

8 64
9 81
10 100

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы.

Вариант 2

1. Написать программу, которая выводит таблицу квадратов целых положительных чисел, введенных пользователем с клавиатуры. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов нечетных чисел.

Число Квадрат
1 1
3 9
5 25
7 49
9 81

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы.

Вариант 3

1. Написать программу, которая вводит с клавиатуры 5 дробных чисел и вычисляет их среднее арифметическое. Рекомендуемый вид экрана во время работы программы приведен ниже:

Вычисление среднего арифметического последовательности дробных чисел.
После ввода каждого числа нажимайте <Enter>
-> 5.4
-> 7.8
-> 3.0
-> 1.5
-> 2.3
Среднее арифметическое введенной последовательности: 4.00

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 4

1. Написать программу, которая выводит на экран таблицу стоимости, например, яблок в диапазоне от 100 г до 1 кг с шагом 100. Ниже представлен рекомендуемый вид экрана программы во время ее работы (данные, введенные пользователем, выделены полужирным шрифтом).

*Введите цену одного килограмма и нажмите <Enter> (копейки от рублей отделяйте точкой) -> **16.50***
Вес (гр) Стоимость (руб.)
100 1.65
200 3.30
300 4.95
400 6.60
500 8.25
600 9.90
700 11.55
800 13.20
900 14.85
1000 16.50

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 5

1. Написать программу, которая выводит на экран таблицу перевода из градусов Цельсия (C) в градусы по Фаренгейту (F) для значений от 15 до 30 с шагом 1 градус. Перевод осуществляется по формуле F =C*1.8+32

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 6

1. Написать программу, которая выводит таблицу значений функции $y=-2,4x^2+5x-3$ в диапазоне от – 2 до 2 с шагом 0,5. Ниже представлен рекомендуемый вид экрана во время работы программы:

X	*y*
-2	*-22,60*
-1,5	*-15,90*
-1	*-10,40*
-0,5	*-6,10*
0	*-3,00*
0,5	*-1,10*
1	*-0,40*
1,5	*-0,90*
2	*-2,60*

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 7

1. Написать программу, которая вводит с клавиатуры 7 дробных чисел и вычисляет сумму положительных чисел и произведение отрицательных чисел. Рекомендуемый вид экрана во время работы программы приведен ниже:

Вычисление среднего арифметического последовательности дробных чисел.
После ввода каждого числа нажимайте <Enter>

-> 1.4
-> 7.8
-> 3.0
-> -7,6
-> -9,2
-> 1.5
-> 2.3

Сумма положительных чисел равна =
Произведение отрицательных чисел равно =

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 8

1. Написать программу, которая вычисляет среднее арифметическое последовательности дробных чисел, вводимых с клавиатуры. После того, как будет введено последнее число, программа должна вывести минимальное и максимальное число последовательности.

Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Обработка последовательности дробных чисел. Введите количество чисел последовательности -> 5
*Вводите последовательность. После ввода каждого числа нажимайте <Enter> -> **5.4** -> **7.8** -> **3.0** -> **1.5** -> **2.3***
*Количество чисел: **5***
Среднее арифметическое: 4.00
Минимальное число:
Максимальное число:

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 9

1. Написать программу, которая выводит таблицу значений функции $y=-9x^2+2x$ в диапазоне от – 3 до 3 с шагом 1. Ниже представлен рекомендуемый вид экрана во время работы программы:

X	y
-3	-87
-2	-40
-1	-11
0	0
1	-7
2	-32
3	-75

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Вариант 10

1. Написать программу, которая вычисляет произведение последовательности целых чисел, вводимых с клавиатуры. После того, как будет введено последнее число, программа должна вывести минимальное и максимальное число последовательности. Количество чисел должно задаваться во время работы программы. Рекомендуемый вид экрана приведен ниже. Данные, введенные пользователем, выделены полужирным шрифтом.

Обработка последовательности дробных чисел. Введите количество чисел последовательности -> 5
*Вводите последовательность. После ввода каждого числа нажимайте <Enter> -> **5** -> **7** -> **3** -> **1** -> **2***
*Количество чисел: **5***
Произведение: 210
Минимальное число:
Максимальное число:

2. Оформить отчет. Отчет должен состоять из кода программы, блок-схемы

Задание 2. Составьте программу, используя оператор Repeat по варианту, предложенному преподавателем.

Вариант 1

Написать программу, вычисляющую произведение положительных чисел, которые вводятся с клавиатуры, используя оператор Repeat. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление произведения положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
*-> **45** -> **23** -> **15***
Введено чисел: 3
Произведение чисел =

Вариант 2

Написать программу, которая определяет максимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности не ограничена). Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Определение максимального числа последовательности
положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
*-> **56***
*-> **75***
*-> **43***
*-> **0***
Максимальное число: 75.

Вариант 3

Написать программу, которая определяет минимальное число из введенной с клавиатуры последовательности положительных чисел (длина последовательности не ограничена). Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Определение максимального числа последовательности положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
*-> **6***
*-> **75***
*-> **3***
*-> **0***
Максимальное число:

Вариант 4

Написать программу, вычисляющую сумму отрицательных чисел, которые вводятся с клавиатуры, используя оператор Repeat. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление произведения положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
*-> **45** -> **23** -> **15**-> **5** -> **2** -> **1***
Сумма чисел =

Вариант 5

Используя цикл Repeat, напишите программу, которая требует ввод пароля, например, числа 111, и если пароль правильный, то выдает на экран сообщение «Вы правильно ввели пароль». Пароль можно вводить три раза.

Вариант 6

Используя цикл Repeat, напишите программу определения идеального веса для взрослых людей по формуле: Ид. Вес = рост – 100.Выход из цикла значение роста = 250

Вариант 7

Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление среднего арифметического последовательности положительных чисел.
Вводите после стрелки числа. Для завершения ввода введите ноль.
-> **4 -> 230-> 15**
Введено чисел:
Сумма чисел:
Среднее арифметическое:

Задание 3. Составьте программы, с использованием оператора While по варианту, предложенному преподавателем.

Программа, ыводит таблицу значений функции для аргумента, изменяющегося в заданных пределах с заданным шагом.

Вариант	**Функция**	
1.	$y = \frac{\sin(3x)}{x^2},$	0<x<10
2.	$y = 1 - x + \frac{x^2}{2} + 5x^4,$	-5<x<6
3.	$y = \cos^2 x + \sin 5x$	-3<x<3
4.	$y = -4\sin(x+5) * \cos x$	-6<x<6
5.	$y = \frac{\sin(\frac{\pi}{2} - 5x)}{x^3}$	-5<x<5
6.	$y = \frac{\sin 2x + \sin 6x}{x^4}$	-3<x<6
7.	$y = \cos^3 x + \sin 10x$	-2<x<5
8.	$y = 2\sqrt{\cos x} - tgx$	-5<x<2
9.	$y = \frac{\sin 10x + x^2 - \cos(6x-2)}{x^4}$	-4<x<2
10.	$y = 2\sqrt{\cos 5x} - \sin x$	-6<x<6

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каких случаях в программе необходимо использовать итерационный цикл, а в каких регулярный цикл?
2. Назовите отличия итерационных циклов и цикла с параметром.
3. Какова структура оператора цикла с параметром? Как выполняется цикл с парметром?
4. Какого типа должны быть пареметр цикла, его начальное и конечное значения в цикле с параметром в языке Pascal?
5. Могут ли параметр цикла, его начальное и конечное значения в цикле с параметром в языке Pascal быть разных типов? Обоснуйте ответ.
6. Чем отличается цикл «До» от цикла «Пока»?
7. Сколько раз повторится итерационный цикл?
8. Какова структура цикла с постропроверкой условия?
9. Какова структура цикла с предпроверкой условия?

ПРАКТИЧЕСКАЯ РАБОТА 6

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ УСЛОЖНЕННОЙ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель работы: научиться составлять программы с использованием вложенных циклов

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Для решения задачи достаточно часто требуется использовать две и более циклические конструкции, одна из которых помещается в тело цикла другой. Такие конструкции называют вложенными циклами. Внутренний и внешний циклы могут быть любыми из трех рассмотренных ранее видов (Практическая работа 5). Правила организации внешнего и внутреннего циклов такие же, как и соответствующих простых операторов. Однако при использовании вложенных циклов необходимо соблюдать следующее условие: внутренний цикл должен полностью укладываться в циклическую часть внешнего цикла.

Цикл, в тело которого мы вкладывали команды, называется **внешним циклом**. А цикл, который мы вложили в тело первого, называется **внутренним или вложенным циклом.**

Принцип работы вложенного цикла таков: на первом проходе, внешний цикл вызывает внутренний, который исполняется до своего завершения, после чего управление передается в тело внешнего цикла. На втором проходе внешний цикл опять вызывает внутренний. И так до тех пор, пока не завершится внешний цикл.

Пример 1. Вывести на экран таблицу умножения (от 1 до 9).

Алгоритм решения

Перебрать во внешнем цикле числа от 1 до 9. Для каждого из них перебрать во внутреннем цикле числа от 1 до 9. Во внутреннем цикле выполнять умножение переменных-счетчиков внешнего и внутреннего циклов. Таким образом на одну итерацию внешнего цикла произойдет девять итераций внутреннего, и сформируется одна строка таблицы умножения. После каждой строки надо перейти на новую. Это делается во внешнем цикле, после того как закончится выполняться внутренний.

Для построения таблицы необходимо использовать форматированный вывод, т.е. задавать ширину столбцов, иначе произойдет сдвиг, т.к. количество цифр в каждой строке различно.

```
var i,j: byte;
begin
    for i:=1 to 9 do begin
        for j:=1 to 9 do
            write(i*j:4);
```

```
        writeln;
    end;
end.
```

Результат работы:

```
 1   2   3   4   5   6   7   8   9
 2   4   6   8  10  12  14  16  18
 3   6   9  12  15  18  21  24  27
 4   8  12  16  20  24  28  32  36
 5  10  15  20  25  30  35  40  45
 6  12  18  24  30  36  42  48  54
 7  14  21  28  35  42  49  56  63
 8  16  24  32  40  48  56  64  72
 9  18  27  36  45  54  63  72  81
```

ЗАДАНИЕ

1. Что будет выведено на экране монитора после выполнения следующего фрагмента программы:

```
a:=1; b:=1;
For i:=0 to n Do
Begin
For j:=1 To b Do
Write ('*') ;
WriteLn;
c:=a+b; a:=b; b:=c;
End;
```

если n=6? Решение какой задачи выражает этот фрагмент программы?

2. Что будет выведено на экране монитора после выполнения следующего фрагмента программы:

```
b: = 0 ;
While a<>0 Do
Begin
b:=b*10+a Mod 10;
a:=a Div 10;
End;
Write (b) ;
```

если a=13305? Решение какой задачи выражает этот фрагмент программы?

3. Выведите на экран числа в следующем виде:

```
7 6 5 4 3 2
6 5 4 3 2
5 4 3 2
4 3 2
3 2
2
```

4. Придумайте задачу, которую можно решить с использованием вложенного цикла

ПРАКТИЧЕСКАЯ РАБОТА 7

ТЕМА: ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ

Цель работы

Изучить принципы работы с одномерными массивами на языке программирования Pascal. Получение навыков применения основных алгоритмов для решения задач с использованием массивов.

Оборудование: ПК, ИСР Pascal ABC

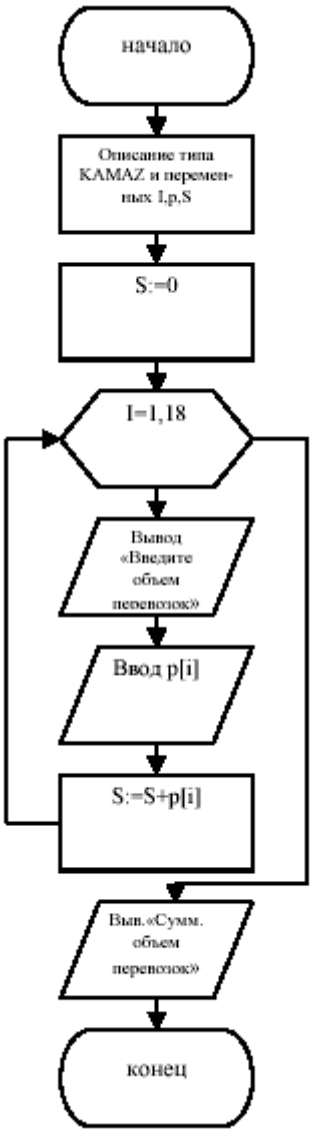
ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Нахождение суммы элементов массива

Задача 1. В автопарке, имеющем 18 машин марки КАМАЗ, каждый из КАМАЗов перевез за день определенный объем груза. Определить суммарный объем перевозок грузов за день. При решении задачи будем использовать тип массива KAMAZ для описания всех КАМАЗов автопарка; переменную P[i] для описания объема груза, перевезенного i-той машиной за день (I меняется от 1 до 18). Блок-схема алгоритма для решения данной задачи будет выглядеть следующим образом (см. **Рис. 1**): Текст следующий вид:

```
Program Pr1;
Uses wincrt;
Type KAMAZ=array [1..18] of  real;
Var
i:integer;
p:KAMAZ;
S:real;
Begin
S:=0;
For i:=1 to 18 do
Begin
Writeln('Введите объем перевозок',I,'-ой машины, т');
Readln(p[i]);
S:=S+p[i]; End;
Writeln('Суммарный объем перевозок S=',S:8:2,'т');
End.
```

Накопление суммы в данном примере будет проводиться по шагам, при вводе значения объема перевозок для очередной машины сумма будет увеличиваться на данную величину. Аналогично реализуется и алгоритм нахождения произведения элементов массива (с заменой начального значения суммы S:=0 на начальное значение произведения S:=1 и с заменой операции сложения элементов массива «+» на операцию умножения «*»).



Нахождение количества элементов массива, удовлетворяющих заданному условию

Задача 2. Известны результаты экзамена по информатике одной группы из 22 студентов. Определить, сколько студентов сдали экзамен на 4 и 5. Один из вариантов решения этой задачи следующий:

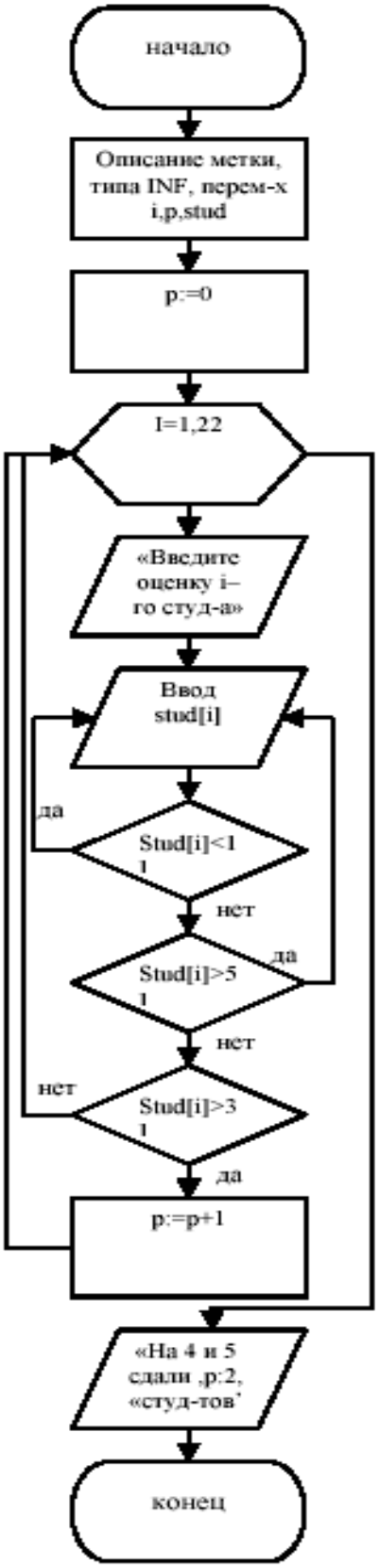


Рис. 2. Блок-схема программы

На Рис. 2 представлена блок-схема алгоритма поставленной задачи.

Текст программы:

```
Program pr3;
Uses wincrt;
Label 1;
Type INF=array[1..22] of integer;
Var
stud:INF;
i,p:integer;
begin
p:=0;
for i:=1 to 22 do
begin
1: writeln('Введите оценку ',i,'-го студента');
readln(stud[i]);
if (stud[i]<1) or (stud[i]>5) then goto 1;
if stud[i]>3 then p:=p+1;
end;
writeln('На 4 и 5 сдали экзамен ',p:2,' студентов');
end.
```

В данной программе для обозначения списка оценок по информатике использовался тип массива INF, для обозначения оценок конкретных студентов – переменная stud. Программа предусматривает проверку
корректности вводимых данных: при попытке ввода несуществующей по пятибалльной системе оценки, программа повторяет ее ввод. Для этого используется оператор перехода GOTO, где имя метки, к которой осуществляется переход (в данном случае 1), описывается в разделе описания меток LABEL.

Сортировка массива по возрастанию

Задача 3. Предположим, известны результаты соревнований по стрельбе, в которых принимали участие 9 человек. Расположить данные результаты в порядке возрастания набранных при стрельбе очков. Алгоритм решения данной задачи является наиболее сложным из приведенных выше примеров и требует использования вложенных циклов.

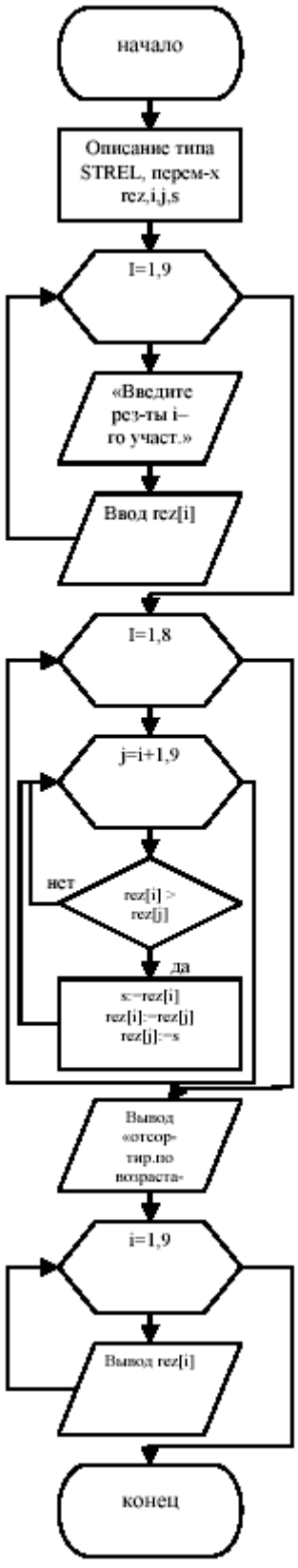


рис. 3. Блок-схема

Один из способов сортировки массивов заключается в следующем. Сначала первый элемент массива в цикле сравнивается по очереди со всеми оставшимися элементами. Если очередной элемент массива меньше по величине, чем первый, то эти элементы переставляются местами. Сравнение продолжается далее уже для обновленного первого элемента. В результате окончания данного цикла будет найден и установлен на первое место самый наименьший элемент массива. Далее продолжается аналогичный процесс уже для оставшихся элементов массива, т.е. второй элемент сравнивается со всеми остальными и, при необходимости, переставляется с ними местами. После определения и установки второго элемента массива, данный процесс продолжается для третьего элемента, четвертого элемента и т.д. Алгоритм завершается, когда сравниваются и упорядочиваются предпоследний и последний из оставшихся элементов массива.

Блок-схема представлена на Рис. 3

Программа реализации изложенного алгоритма может иметь следующий вид:

```
Program pr4;
Uses crt;
Type STREL=array[1..9]of integer;
Var
rez:strel;
i,j,s:integer;
Begin
For i:=1 to 9 do
begin
writeln('Введите результаты ',i,'-го участника');
readln(rez[i]);
end;
for i:=1 to 8 do
for j:=i+1 to 9 do
if rez[i]>rez[j] then
begin
s:=rez[j];
rez[j]:=rez[i];
rez[i]:=s;
end;
writeln('Отсортированные по возрастанию результаты:');
for i:=1 to 9 do write (rez[i]:5,' ');
end.
```

Здесь STREL – тип массива результатов стрельбы участников, rez[i] – переменная для описания результатов i-го участника (i меняется от 1 до 9).

Вспомогательная переменная s используется для перестановки местами элементов массива.

ВАРИАНТЫ ЗАДАНИЙ

Исходные данные необходимо оформить в виде массива. При выполнении задания ввод исходных данных и вывод результатов сопровождать комментариями (какие данные нужно ввести и что получается в результате). Составить блок-схему программы. Оформить отчет

1. Подсчитать среднемесячную зарплату сотрудника предприятия.
2. Дан объем продукции, выпущенной заводом за год (помесячно). Найти наименьший объем. В качестве результата вывести номер месяца и объем выпущенной продукции.
3. Курс доллара в течение года менялся в диапазоне от 28руб. до 30руб. Найти наибольшее значение курса доллара. В качестве результата вывести номер месяца и значение курса доллара.
4. Известен месячный план выпуска некоторой продукции и объемы выпущенной этой продукции заводом за год (помесячно). Определить, когда завод не выполнил план. Результат получить в виде: номер месяца и объем выпущенной продукции.
5. Даны результаты сдачи экзамена по информатике группы студентов (в группе 20 студентов). Подсчитать количество студентов, не сдавших экзамен.
6. Известна среднемесячная зарплата 10 сотрудников отдела. Расположить данные в порядке убывания.
7. Известен годовой процент на вклад с капитализацией (начисление процентов ежемесячно). Определить, сколько денег получит вкладчик в конце года, если на 1 января сумма вклада составляла 1500руб. в качестве результата вывести сумму вклада на конец каждого месяца.
8. Известны данные по продаже компьютеров в течение недели. Найти общее количество проданных компьютеров.
9. Известны данные по продаже компьютеров в течение недели. Расположить эти данные в порядке возрастания.
10. Известен месячный план выпуска некоторой продукции и объемы выпущенной продукции заводом за год (помесячно). Определить месяц, в котором было максимальное отклонение от плана. В качестве результата вывести номер месяца и отклонение.
11. Известен месячный план выпуска некоторой продукции и объемы выпущенной продукции заводом за год (помесячно). Определить, был ли выполнен годовой план.
12. Даны результаты сдачи экзамена по информатике группы студентов (в группе 20 студентов). Подсчитать количество студентов, сдавших экзамен без троек.
13. Известен месячный план выпуска некоторой продукции и объемы выпущенной этой продукции заводом за год (помесячно). Определить, когда завод перевыполнил план. Результат получить в виде: номер месяца и объем продукции, выпущенной сверх плана.
14. Подсчитать среднемесячную зарплату сотрудника предприятия и найти зарплату, которая наиболее близка к средней. В качестве результата вывести среднюю зарплату, наиболее близкую и ее номер в массиве.
15. Даны результаты сдачи экзамена по информатике группы из 15 студентов. Подсчитать количество студентов, не сдавших экзамен, в численном и в процентном соотношении.

ПРАКТИЧЕСКАЯ РАБОТА 8

ТЕМА: ОБРАБОТКА МНОГОМЕРНЫХ МАССИВОВ

Цель работы:

Изучить принципы работы с многомерными массивами на языке программирования Pascal. Получение навыков применения основных алгоритмов для решения задач с использованием массивов.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Исходные данные для решения многих задач удобно представить в виде таблицы. Например, результат производственной деятельности заводов некоторой фирмы можно представить в виде таблицы (см.Таблица 1)

Колонки и строки таблицы, как правило, содержат однородную информацию, если использовать терминологию Pascal – данные одинакового типа. Поэтому в программе для хранения и обработки табличных данных можно использовать совокупность одномерных массивов. Так, приведенная выше таблица может быть представлена как совокупность одномерных массивов следующим образом:

```
Zavod1:array[1..4] of integer;
Zavod2:array[1..4] of integer;
Zavod3:array[1..4] of integer;
```

Каждый из приведенных массивов может хранить информацию о количестве продукции, выпущенной одним заводом.

Возможно и такое представление:

```
product1:array[1..3] of integer;
product2:array[1..3] of integer;
product3:array[1..3] of integer;
product4:array[1..3] of integer;
```

Таблица 1

	Продукт1	Продукт2	Продукт3	Продукт4
Завод1	1500	14000	15	125
Завод2	1380	15600	25	140
Завод3	2500	13000	8	165

В этом случае массив предназначен для хранения информации о количестве продукции одного наименования, произведенной каждым заводом. Помимо совокупности одномерных массивов, таблица может быть представлена как двумерный массив. В общем виде описание двумерного массива выглядит следующим образом:

*Имя:***array**[*нижняя_граница_индекса1..верхняя_граница_индекса1,нижняя_граница_индекса 2..верхняя_граница_индекса2*] **of** *тип*

где *Имя* – имя массива;

array – ключевое слово, показывающее, что объявляемый элемент данных является массивом;

нижняя_граница_индекса1, верхняя_граница_индекса1, нижняя_граница_индекса2, верхняя_граница_индекса2 – целые константы, определяющие диапазоны изменения индексов и, следовательно, число элементов массива;
тип – тип элементов массива.

Приведенная выше таблица (см. Таблица 1) может быть представлена в виде двумерного массива следующим образом:
Product:**array**[1..3,1..4] **of** integer;

Чтобы использовать элемент массива, нужно указать имя массива и индексы элемента. Первый индекс обычно соответствует номеру строки таблицы, второй – номеру колонки. Так, элемент product[2,3] содержит число продуктов третьего наименования, выпущенных вторым заводом.

Двумерные массивы, в которых диапазоны индексов начинаются с 1, также называются иногда *матрицами*. Размерность матрицы определяется как M*N, где M – число строк в матрице, N – число столбцов. Если число строк матрицы равняется числу столбцов, то матрицы такого вида называются *квадратными*.

Элементы квадратной матрицы вида B[1,1], B[2,2]< B[3,3] составляют главную диагональ матрицы. Иногда вводят понятие побочной диагонали квадратной матрицы, которую составляют элементы B[1,N], B[2,N-1], B[3,N-2], .. B[N,1], где N – число строк (столбцов) матрицы.

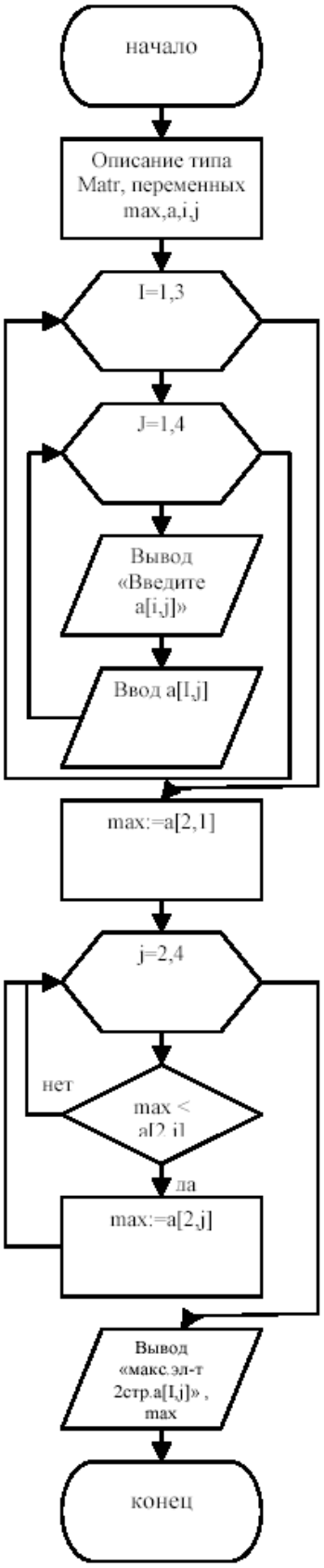
Приведем еще примеры описания двумерных массивов в Pascal-программах:
Type MATR=**array**[1..4,1..5] **of** integer;
Type B= **array**[2..9,0..6] **of** real;
Type C= **array**[-1..4,-1..4] **of** char.
Также допускается указание имени другого типа массива в качестве типа элементов массива, например:
Type VEC= **array**[1..4] **of** real;
MAS= **array**[1..5] **of** vec.

В результате приведенного выше описания тип массива MAS будет объявлен как тип двумерного массива, первый индекс которого будет меняться от 1 до 5, а второй индекс – от 1 до 4, т.е. размерность массива составит 5*4 элементов.

При вводе и выводе значений элементов двумерных массивов используются вложенные циклы, в которых внешний оператор цикла, как правило, задает изменение строк массива, внутренний оператор цикла – изменение столбцов.

ПРИМЕРЫ ЗАДАЧ

1. Нахождение наибольшего элемента в заданной строке.



.

Рис. 1. Блок-схема программы

Пусть задана матрица А из действительных чисел размера 3х4. Найти наибольший элемент во второй строке данной матрицы. Блок-схема алгоритма решения данной задачи представлена на Рис. 1

```
Program Max_str;
Uses crt;
Type Matr=array[1..3,1..4] of real;
Var max:real;
a:Matr;
i,j:integer;
begin
for i:=1 to 3 do
for j:=1 to 4 do
begin
writeln('Введите элемент a[',i,',',j,']');
readln(a[i,j]);
end;
max:=a[2,1];
for j:=2 to 4 do
if max<a[2,j] then max:=a[2,j];
writeln('Наибольший элемент второй строки=',max:8:2);
end.
```

Данная программа представляет собой реализацию алгоритма нахождения наибольшего элемента вектора, полученного путем фиксирования одного из индексов двумерного массива.

2. Нахождение элементов массива, удовлетворяющих заданному условию.

Известны результаты 5 студентов по итогам экзаменов по химии и информатике. Найти фамилии студентов, сдавших оба экзамена на отлично. Для решения поставленной задачи может быть использована следующая программа (ее блок-схема представлена на рис. 2.

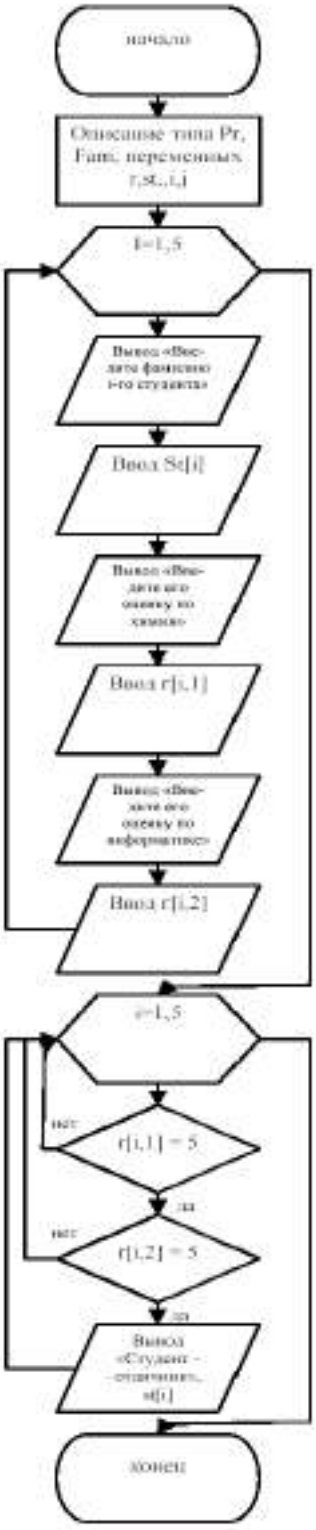


Рис. 2. Блок-схема программы

```
Program Sessia;
type PR=array [1..5,1..2]of integer;
Fam=array[1..5]of string[10];
var r:pr;
st:fam;
i,j:integer;
begin
for i:=1 to 5 do
begin
writeln('Введите фамилию ',i,'-го студента');
readln(st[i]);
writeln('Введите оценку данного студента по химии (от 2 до 5)');
readln(r[i,1]);
writeln('Введите оценку данного студента по информатике (от 2 до 5)');
readln(r[i,2]);
end;
for i:=1 to 5 do
if (r[i,1]=5) and (r[i,2]=5) then
writeln('Студент-отличник - ',st[i]);
end.
```

В данной программе для хранения фамилий студентов используется одномерный строковый массив st типа Fam, для хранения оценок студентов – двумерный целочисленный массив r типа PR, причем первый столбец матрицы r используется для хранения результатов экзамена по химии, второй
столбец – экзамена по информатике. Если у некоторого студента оценки за оба экзамена составили 5 баллов, то его фамилия будет выведена на экран с сообщением «Студент-отличник».

3. Нахождение сумм элементов строк матриц

Рассмотрим задачу нахождения сумм элементов строк матрицы на примере задачи подсчета итогов футбольного чемпионата. Пусть задана таблица результатов игр 5 команд футбольного чемпионата размером 5x5. На диагонали таблицы стоят значения 0, другие элементы таблицы равны 0, 1 или 2, где 0 баллов соответствует проигрышу команды в игре, 1 балл – ничьей, 2 балла – выигрышу. Определить сумму баллов каждой команды по результатам чемпионата.

Легко заметить, что для построения матрицы R результатов игр достаточно ввести лишь стоящую выше (или ниже) главной диагонали половину матрицы, т.к. результаты остальных игр могут быть рассчитаны из известного соотношения: если, например, первая команда обыграла вторую, то элемент R[1,2]=2, а элемент R[2,1]=2-R[1,2]=0; аналогично, если вторая команда сыграла в ничью с третьей, то R[2,3]=1, R[3,2]=2-R[2,3]=1. Таким образом, нетрудно получить вид взаимосвязи элементов матрицы: R[i,j]+R[j,i]=2, где i и j меняются от 1 до 5 (кроме элементов главной диагонали). На главной диагонали матрицы R по условию задачи всегда стоят числа 0.

```
Program foot;
Type tab=array[1..5,1..5] of integer;
Var r:tab; i,j,s:integer;
begin
{ввод стоящих выше диагонали элементов матрицы}
for i:=1 to 4 do
for j:=i+1 to 5 do
begin
        writeln (‘Введите результат игры ‘,i,’–й команды с ’,j,’ –й: 0, 1 или 2 балла’);
        readln(r[i,j]);
end;
{заполнение стоящих на диагонали элементов нулями}
for i:=1 to 5 do r[i,i]:=0;
{вычисление стоящих ниже диагонали элементов матрицы}
for i:=2 to 5 do
for j:=1 to i-1 do r[i,j]:=2-r[j,i];
{вывод на экран матрицы результатов игр}
writeln(‘Таблица чемпионата’);
for i:=1 to 5 do
begin
        for j:=1 to 5 do write(r[i,j]:4);
        writeln;
end;
{вычисление сумм элементов строк матрицы}
for i:=1 to 5 do
begin
        s:=0;
        for j:=1 to 5 do s:=s+r[i,j];
        writeln(i,‘-ая команда набрала ’,s:3,’ очков’);
end;
end.
```

ЗАДАНИЯ

Исходные данные необходимо оформить в виде двумерного массива, в части заданий использовать дополнительно и одномерные массивы. При выполнении задания ввод исходных данных и вывод результатов сопровождать комментариями (какие данные нужно ввести и что получается в результате).

1. Известен план выпуска компьютеров и количество выпущенных компьютеров тремя фирмами за шесть месяцев. Определить для каждой фирмы, был ли выполнен план по итогам шести месяцев.
2. Известно количество сделанных столов тремя фабриками за два квартала. Определить максимальное количество выпущенных столов. В качестве результата вывести месяц, в котором это было, и название фабрики.
3. Известен план выпуска компьютеров и количество выпущенных компьютеров тремя фирмами за три месяца. Определить, в каком месяце не был выполнен план третьей фирмой.
4. Известен план выпуска компьютеров и количество выпущенных компьютеров тремя фирмами за шесть месяцев. Определить для каждой фирмы количество месяцев, когда план был перевыполнен.
5. Известна заработная плата, полученная 5 сотрудниками отдела в течение года. Определить максимальную заработную плату. В качестве результата вывести фамилию и размер заработной платы.
6. Известно количество выпущенной продукции тремя заводами за первый квартал (помесячно). Найти среднемесячное количество выпущенной продукции для каждого завода.
7. Известно количество выпущенной продукции тремя заводами за первый квартал (помесячно). Найти среднемесячное количество выпущенной продукции по всем заводам.
8. Известны результаты сдачи трех экзаменов пятью студентами. Найти фамилии студентов, не сдавших оба экзамена.
9. Известно количество сделанных столов тремя фабриками за два квартала. Определить, какая фабрика выпустила максимальное количество столов по итогам шести месяцев.
10. Известна заработная плата, полученная 10 сотрудниками отдела в течение года. Определить среднемесячную зарплату по отделу.
11. Известны результаты сдачи двух экзаменов семью студентами. Найти фамилии студентов, не сдавших хотя бы один экзамен.
12. Известно количество сделанных столов тремя фабриками за два квартала. Определить, какая фабрика выпустила минимальное количество столов по итогам первых трех месяцев.
13. Известны результаты сдачи трех экзаменов десятью студентами. Найти средний балл каждого студента и общий средний балл. Точность среднего балла – два знака после запятой.
14. Известно количество сделанных столов тремя фабриками за два квартала. Определить, какая фабрика выпустила максимальное количество столов по итогам второго квартала.
15. Известны результаты сдачи двух экзаменов десятью студентами. Определить фамилии студентов, сдавших экзамены без троек.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимают под массивом данных?
2. Что называют размерностью массива?
3. Что понимают под индексом элемента массива?
4. Какой массив называется одномерным?

5. Приведите примеры одномерных массивов.
6. Как описываются одномерные массивы на языке PASCAL?
7. Как задается диапазон изменения индексов массива?
8. Как обозначаются индексы массивов на языке PASCAL?
9. Какие стандартные алгоритмы по работе с одномерными массивами Вы
10. знаете?
11. Поясните понятия двумерного массива, матрицы.
12. Что обозначают индексы матрицы?
13. Сколько элементов в матрице из 7 строк и 9 столбцов?
14. Дайте понятие квадратной матрицы, диагоналей квадратной матрицы.
15. Приведите пример описания двумерных массивов на языке PASCAL.
16. Поясните порядок использования вложенных циклов при вводе элементов
17. двумерного массива.

ПРАКТИЧЕСКАЯ РАБОТА 9

ТЕМА: ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ФУНКЦИЙ И ПРОЦЕДУР ДЛЯ РАБОТЫ СО СТРОКАМИ

Цель работы

Целью работы является приобретение навыков алгоритмизации и программирования задач, оперирующих строковыми типами данных:

- ввод и вывод строковых данных;
- обработка строковых данных;
- использование стандартных процедур и функций языка Pascal для обработки строковых данных.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Обработка строковых данных является необходимым элементом программ, работающих с текстами. К таким программам относятся программы лингвистического анализа текстов, текстовые редакторы, программы, работающие в диалоговом (интерактивном) режиме, программы, содержащие текстовые пояснения результатов своей работы.

Строка – последовательность символов (от 0 до 255), заключенная в апострофы. При составлении программ используются строковые константы и строковые переменные.

1. Определение через описание типа в разделе описания типов.

Формат

```
TYPE
<имя типа> = STRING [максимальная длина строки];
VAR
<идентификатор1, идентификатор2,…> : <имя типа>;
```

где

STRING – зарезервированное слова (строка);

Максимальная длина строки – наибольшее допустимое количество символов переменной данного типа (**.** 255**).**

Например,

```
TYPE
FLO = STRING [130];
FTK = STRING; {По умолчанию длина строки равна 255}
VAR
ST1 : FLO;
ST2, ST3 : FTK;
```

2. Определение непосредственно в разделе описания переменных

Формат

```
VAR
ST4, ST5 : STRING [60];
ST6, ST7 : STRING;
```

ПРИМЕР ЗАДАЧИ

Разработать программу, удаляющую из вводимой с клавиатуры строки пробелы между словами и записывающую в массив N длину (число символов) каждого слова. Длина текста – не более 80 символов. Число слов – не более 10. Наличие более одного символа ‘пробел’ подряд свидетельствует о конце строки.

Используемые в программе идентификаторы приведены в таблице 1.

Таблица 1

Обозначения	Тип данных	Примечание
A	STRING	Исходный текст, символьные данные
K	INTEGER	Количество символов в слове
L	INTEGER	Порядковый номер слова
I	INTEGER	Параметр цикла
A[I]		Текущий символ исходного текста
N	ARRAY [1…10] OF INTEGER	Массив, содержащий значения длины каждого слова исходного текста
N[L]		Значение длины слова номер L
J	INTEGER	Параметр цикла, используемого для перемещения всех следующих символов исходного текста на одну позицию влево после того, как обработано очередное слова.
PR	INTEGER	Переменная для управления повторной работой программы
OTVET	BYTE	Переменная для управления началом обработки введенной строки

Листинг программы

```
Program Prim1;
Uses Crt;
Label 4;
VAR
N: ARRAY [1..10] OF INTEGER;
I, J, K, L: INTEGER;
A: STRING [80];
PR, OTVET:BYTE;
BEGIN
CLRSCR;
REPEAT
                REPEAT
                        WRITELN (' Введите через пробел');
                        READLN (A);
                        WRITELN('Исходная строка');
                        WRITELN(A);
                        WRITELN ('Работаем дальше? 1 -да,0 -нет');
                        READLN (OTVET);
                UNTIL OTVET=1;
                        K:=0;
                        L:=0;
                        PR:=0;
                        FOR I:=1 TO length(a) DO
                                IF (A[I]= ' ') THEN
                                                BEGIN
                                                        L:=L+1;
                                                        N[L]:=K;
                                                        IF (A[I+1]=' ')THEN GOTO 4;
                                                        FOR J:=i TO length(a) DO
                                                        A[J]:= A[J+1];
                                                        K:=1
                                                END
                                        ELSE
                                                begin
                                                        K:=K+1;
                                                        N[L+1]:= K-l;
                                                end;
                        4:WRITELN ('Результирующая строка');
```

```
                    WRITELN (A);
                    WRITELN ('№ слова число букв');
                    FOR I:=1 TO L+1 DO
                    WRITELN (' N[',I,'] =',N[I]:6);
                    WRITELN('Обработать еще одну строку? 1 –да 0 -нет');
                    READLN(PR);
UNTIL PR=0
END.
```

Протокол работы программы

Протокол работы программы показан на рис. 1.

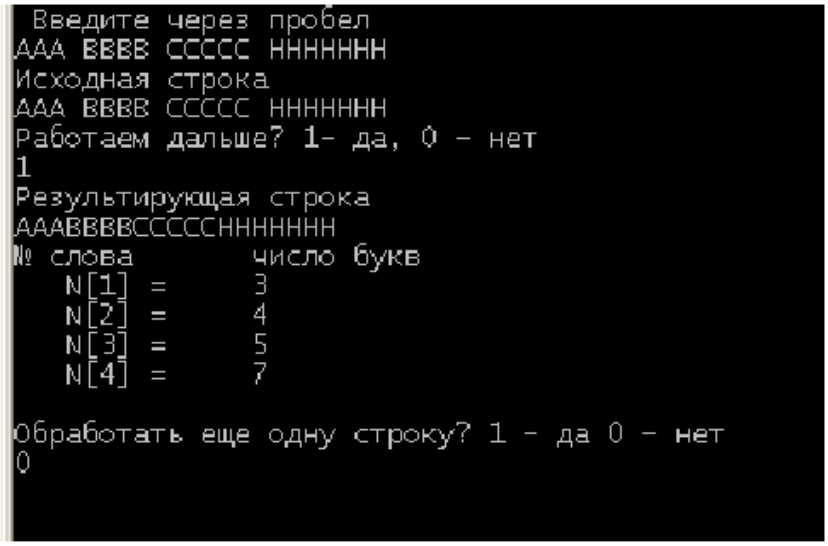


Рис. 1. Протокол работы программы

ЗАДАНИЕ

1. В заданном тексте удалить символ ‘,’ и подсчитать число удаленных символов. Предусмотреть возможность задания с клавиатуры удаляемого символа.
2. В заданном тексте заменить словосочетание «свернутые обороты» на словосочетание «разделенные обороты» и подсчитать число произведенных замен.
3. Из заданного предложения выбрать и вывести на экран только те символы, которые встречаются в нем только один раз (в том порядке, в каком они встречаются в тексте)
4. В заданном предложении найти самое длинное и самое короткое слова и подсчитать, на сколько больше символов в самом длинном слове.
5. Проверить, встречается ли в заданном предложении словосочетание «Остаток счета».
6. Проверить, встречается ли в заданном предложении словосочетание «Сальдо счета».
7. Для каждого слова заданного предложения указать долю согласных букв. Определить слово, в котором доля согласных максимальна.
8. Из заданного текста выбрать цифры и записать в массив N, а буквы- в массив B. Все остальные символы записать в массив S.
9. Удалить из заданного текста все пробелы, подсчитать длину получившегося текста и число удаленных пробелов.
10. Для заданного предложения указать слова, в котором доля гласных букв максимальна.
11. Отредактировать предложение, удаляя из него лишние пробелы, оставив только по одному пробелу между словами. Подсчитать число удаленных пробелов.

12. Проверить, имеется ли в заданном тексте баланс открывающихся и закрывающихся круглых скобок.
13. Дана последовательность из 10 слов. Вывести слова в обратном порядке.
14. Дана последовательность из 10 слов. Вывести все слова, входящие в эту последовательность по одному разу.
15. Дана последовательность из 8 слов. Вывести входящие в эту последовательность слова, расположив их по алфавиту.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое строка?
2. Для чего в программировании используются строки?
3. Как объявляется строка в программе?
4. Как обозначается элемент строки?
5. Назовите процедуры для обработки строк
6. Назовите функции для обработки строк

ПРАКТИЧЕСКАЯ РАБОТА 10

ТЕМА: СОСТАВЛЕНИЕ ПРОГРАММ СО СТРУКТУРИРОВАННЫМ ТИПОМ ДАННЫХ «МНОЖЕСТВО»

Цель работы: организовывать программы и использованием структурированного типа данных «множество».

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В Pascal отсутствуют средства ввода-вывода элементов множества, поэтому работу программы необходимо проверять, выполняя ее в пошаговом режиме и отслеживая изменения значений переменных в окне просмотра:

1. С помощью команды ***Debug/Add*** watch задать переменные, значения которых необходимо наблюдать.
2. Открыть окно наблюдаемых значений, выбрав команду ***Debug/Watch***, а окно редактора уменьшить так, чтобы окна не перекрывали друг друга.
3. Установить курсор на точку программы, до которой она выполняется правильно
4. Запустить программу до этой точки, выбрав команду ***Run/Go to cursor (F4)***, а затем выполнять ее по шагам с помощью команды ***Run/Step over (F8),*** наблюдая в окне значения переменных.
5. Для прерывания процесса отладки выбрать команду ***Run/Program reset (Ctrl+F2)***

Пример

```
Program Dem_Mno;        {Демонстрация операций над множествами}
type
Digits=set of 0..9;
var
d1, d2,d3,d: digits;
begin
d1:=[2, 4,6,8];         {Заполнение множеств}
d2:=[0..3,5];
d3:=[1,3,5,7,9];

d:=d1+d2;               {Объединение множеств d1и d2}
d:=d+d3;                {Объединение множеств d и d3}
d:=d-d2;                {Разность множеств d и d2}
d:=d*d1;                {Пересечение множеств d и d1}
end.
```

ЗАДАНИЯ

1. Опишите множество Pr(1..20) и поместите в него все простые числа в диапазоне 1..20. Составьте блок-схему.
2. Опишите множество Alf(‘а’..’я’) и поместите в него гласные буквы. Составьте блок-схему.
3. Опишите множества M1(1,2) и M2(2,1). Сравните множества M1 и M2 на равенство. Составьте блок-схему.

4. Опишите множества M1(‘a’, ‘b’) M2(‘b’, ’a’, ‘c’). Сравните два этих множества на неравенство. Составьте блок-схему.
5. Опишите множества M1(‘a’, ‘b’,’c’) M2(‘a’, ‘c’). Сравните два этих множества с использованием операции >=. Составьте блок-схему.
6. Опишите множества M1(1, 2, 3) M2(1, 2, 3, 4). Сравните два этих множества с использованием операции <=. Составьте блок-схему.
7. Опишите множества M1(1, 2) M2(5, 6). Получите результирующее множество M3=M1-M2. Определите содержится в M3 элемент 7. Составьте блок-схему.
8. Опишите множества M1(1, 2, 3,4) M2(3, 4, 1). Получите результирующее множество M3=M1-M2. Определите содержится в M3 элемент 7. Составьте блок-схему.
9. Опишите множества M1(1, 2, 3) M2(1, 4, 2, 5). Получите результирующее множество M3=M1*M2. Определите содержится в M3 элемент 2. Составьте блок-схему.
10. Опишите множества M1(1, 2) M2(5, 6). Получите результирующее множество M3=M1+M2. Определите содержится в M3 элемент 2. Составьте блок-схему.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое множество?
2. Как в Паскале организовывается работа с множеством?
3. Перечислите операции сравнения множества
4. Как описать множество в программе?
5. С помощью какой функции определить содержится ли элемент в множестве?

ПРАКТИЧЕСКАЯ РАБОТА 11

ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕ ПРОЦЕДУР. ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ

Цели работы:

- Научиться описывать процедуры и функции в программе;
- Научиться задавать фактические и формальные параметры;
- Научиться организовывать вызов процедуры и функции в программе

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Подпрограмма — это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы. Каждая подпрограмма определяется уникальным именем.

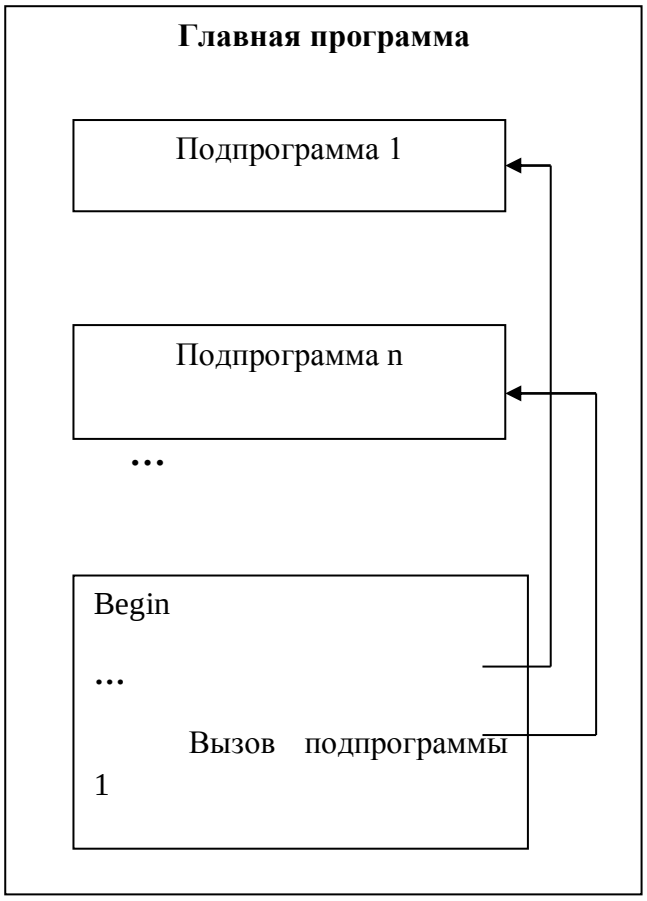


Рис. 1 Структура программы с подпрограммами

Различают два вида подпрограмм — это процедуры и функции.

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

Процедура — это независимая именованная часть программы, которую можно вызвать по имени для выполнения определённой в ней последовательности действий. Функция отличается от процедуры тем, что возвращает результат указанного при её описании типа. Вызов функции может осуществляться из выражения, где имя функции используется в качестве операнда.

Заголовок процедуры имеет вид: PROCEDURE <имя> [(<сп. ф. п .>)] ;
Заголовок функции: FUNCTION <имя> [(<сп.ф.п.>)] : <тип>;
Здесь <имя> - имя подпрограммы (правильный идентификатор);
<сп.ф.п.> - список формальных параметров;
<тип> - тип возвращаемого функцией результата.

Описание, определение и вызов процедур

Описание процедуры производится в разделе описаний основной программы. Любая процедура оформляется аналогично программе, может содержать заголовок, разделы описаний и операторов. Синтаксис заголовка процедуры:
Procedure <Name>(<Список формальных параметров>);
{Раздел описаний}
Begin …{Раздел операторов процедуры}End;
где
Procedure - служебное слово;
Name - произвольный идентификатор, определяющий имя процедуры.

```
Procedure MyProc (A,B,C: Real; var X1,X2: Real);
Begin
        WriteLn('A=',A, ' B=', B, 'C=', C);
        X1:=A+B;
        X2:=A*B-C
End;
```

Разделы описаний процедуры подобно основной программе могут содержать разделы описания меток (Label), констант (Const), типов (Type), переменных (Var) и раздел процедур и функций. Раздел операторов помещается после служебного слова Begin и заканчивается служебным словом End, после End ставится " ; ".

В основной программе процедуры располагают перед разделом операторов (телом программы) основной программы.

Формальные параметры — это переменные, посредством которых передаются данные из места вызова процедуры в её тело, либо из процедуры в места вызова. Список формальных параметров может отсутствовать, при этом символ " ; " ставится сразу за именем процедуры и данные из места вызова процедуры в её тело не передаются.

Для вызова процедуры на исполнение к ней необходимо обратится.

Вызов процедуры производится указанием имени процедуры и списком фактических параметров:

Name(<Список фактических параметров>);

MyProc(K, L+M, 12, Y1, Y2);

Выполнение оператора вызова процедуры состоит в том, что все формальные параметры заменяются соответствующими фактическими. После выполнения процедуры

происходит передача управления в основную программу, т.е. начинает выполняться оператор, следующий за оператором вызова процедуры.

Фактические параметры — это переменные (или значения заданные явно), которые передаются в процедуры на место формальных параметров. Если в вызываемой процедуре отсутствует список формальных параметров, то список фактических параметров тоже отсутствует.

Количество фактических параметров должно соответствовать количеству формальных параметров; соответствующие фактические и формальные параметры должны совпадать по порядку записи и по типу данных.

Описание, определение и вызов функции

Оформляется функция аналогично процедуре. Отличительной особенностью функции является то, что она возвращает только один результат выполнения. Этот результат обозначается именем функции и возвращается (передается) в основную программу (место вызова). Функция состоит из заголовка, раздела описаний и раздела операторов.

```
Function <Name>(<Список формальных параметров>):<Type>;
… {Раздел описаний}
Begin
…{Раздел операторов процедуры}
Name:=<выражение соответствующего типа;
…
End;
```

где Function - служебное слово; Name - произвольный идентификатор, определяющий имя функции. В отличии от процедур в разделе операторов тела функции обязательно должен быть хотя бы один оператор присвоения имени функции выражения или значения соответствующего типа. После работы функции результат присваивается имени функции.

Таким образом, алгоритм можно оформить в виде функции в том случае, если в качестве результата получается одно единственное значение. Для вызова функции достаточно указать ее имя (с фактическими параметрами) в любом выражении, где тип результата функции будет приемлем. Имя функции можно использовать в арифметических выражениях и других командах.

Пример 1. Разработать функцию, определяющую по двум катетам гипотенузу прямоугольного треугольника.

```
Function Gepoten(a,b:real):real;
Begin
 Gepoten:=Sqrt(Sqr(a)+Sqr(b))
End;
```

Вызов функции из основной программы может выглядеть следующим образом: z:=Gepoten(x, y); {z присваивается значение гипотенузы}

или WriteLn('Значение гипотенузы', Gepoten(x, y));

Передача параметров в подпрограммы

Передача параметров в подпрограмму может осуществляться несколькими способами. Параметры процедур и функций могут быть следующих видов: параметры-значения, параметры-переменные, параметры-константы и не типизированные параметры.

Группа параметров без предшествующего ключевого слова является списком ***параметров-значений***

Группа параметров, перед которыми следует ключевое слово Const и за которыми следует тип, является списком ***параметров-констант***.

Группа параметров, перед которыми стоит ключевое слово Var и за которыми следует тип, является списком типизированных ***параметров-переменных.***

Группа параметров, перед которыми стоит ключевое слово Var или Const за которыми не следует тип, является списком не типизированных параметров-переменных.

Передача параметров по значению

При передаче параметров по значению в формальный параметр передаётся копия значения соответствующего фактического параметра. Примерами параметров-значений служат параметры A, B и C в процедуре с заголовком MyProc. В этом случае фактическим параметром, соответствующим A, либо B, либо C, может быть любое выражение или переменная типа Real. Для параметров-значений компилятор при вызове процедуры выделяет место в сегменте стека (специальная область оперативной памяти) для каждого формального параметра, вычисляет значение фактического параметра и передаёт его в ячейку, соответствующую формальному параметру, выполняет тело процедуры. После завершения работы процедуры, формальные параметры уничтожаются, а фактические остаются неизменными (по значению).

Формальный параметр-значение обрабатывается, как локальная по отношению к процедуре или функции переменная, за исключением того, что он получает свое начальное значение из соответствующего фактического параметра при активизации процедуры или функции.

Соответствующее фактическое значение параметра-значения должно быть выражением и его значение не должно иметь файловый тип или какой-либо структурный тип, содержащий в себе файловый тип. Фактический параметр должен иметь тип, совместимый по присваиванию с типом формального параметра-значения. Если параметр имеет строковый тип, то формальный параметр будет иметь атрибут размера, равный 255.

Приступая к решению задач этого раздела, следует вспомнить, что:

- для передачи информации в процедуру следует использовать параметры, а не глобальные переменные, т. е. объявленные вне процедуры;
- тип каждого фактического параметра (константы или переменной) в инструкции вызова процедуры должен соответствовать типу соответствующего формального параметра, указанного при объявлении функции;
- если в инструкции объявления процедуры перед именем формального параметра нет слова var, то в качестве формального параметра в инструкции вызова процедуры можно использовать константу или переменную соответствующего типа. Если слово var присутствует в инструкции, то формальным параметром можно назначить только переменную;
- если аргумент процедуры применяется для возврата результата в программу, вызвавшую эту процедуру, то перед именем аргумента нужно поставить слово var.

Пример 1

Программа вычисления степени по вводимым пользователем значением числа и показателя степени.

```
program Func;
uses crt;
var
a,z,r: integer;
m:integer;
{Функция вычисления степени, N, X – формальные параметры}
function Stepen(n:integer; x:integer):integer;
    var
    i:integer;
    y: integer;
```

```
    begin
    y:=1;
    for I:=1 to n do        {Цикл вычисления n-ой степени числа x}
    y:=y*x;
    stepen:=y               {Присваивание функции результата вычисления степени}
    end;                    {Конец функции}
Begin
clrscr;
writeln('vvedite znachenie chisla a i pokazatel stepeni m');
readln(a);
readln(m);
z:=Stepen(m,a);
Writeln('z=', z:3);
end.
```

Пример № 2

Даны два натуральных числа a и b. Требуется определить наибольший общий делитель трех величин. Определить наибольший общий делитель трех величин a+b, |a-b|, a*b

Идея решения состоит в следующем математическом факте: НОД(x, y, z) = НОД(НОД(x,y),z)

```
Program NOD;
Uses crt;
var
a,b,c: integer;

Procedure Evklid (M,N: integer; Var k: integer); {m,n – формальные параметры (параметры-аргументы, к – параметр-результат)}
    Begin
    while m<>n Do
    If m>n
    then m:=m-n else n:=n-m;
    k:=m
    End;

Begin
clrscr;
write('a=');
readln(a);
Write('b=');
readln(b);
Evklid(a+b, ABC(a-b), a*b);
Evklid(c, a*b, c);
Writeln('NOD=',c)
End.
```

ЗАДАНИЕ

Напишите программу на языке программирования, используя процедуры и функции, по варианту, предложенному преподавателем.

Вариант 1

Дана целочисленная квадратная матрица 4х4. Определить:

1) произведение элементов во второй строке (оформить в виде функции)
2) сумму элементов главной диагонали (оформить в виде процедуры)
3) составить блок-схемы

Вариант 2

Дана целочисленная матрица размером 4х3. Определить:

1) Количество нулевых элементов в 4 строке. (Оформить в виде функции).
2) Максимальный элемент матрицы (оформить в виде процедуры)
3) составить блок-схемы

Вариант 3

Дана целочисленная прямоугольная матрица. Определить:

1) Сумму положительных элементов матрицы (оформить в виде функции)
2) Минимальный элемент матрицы (оформить в виде процедуры)
3) составить блок-схемы

Вариант 4

Дана целочисленная квадратная матрица 3х3. Определить:

1) количество отрицательных элементов матрицы (оформить в виде функции)
2) количество положительных элементов в 3 строке (оформить в виде процедуры)
3) составить блок-схемы

Вариант 5

1. Сформировать одномерный массив B с помощью генератора случайных чисел в интервале от 0 до 50 (оформить в виде процедуры).
2. Подсчитать для сформированного массива B сумму положительных элементов (оформить в виде функции).
3. Составить блок-схемы

Вариант 6

1. Сформировать многомерный массив размерностью 3х3, ввод элементов осуществлять с клавиатуры (оформить в виде процедуры).
2. Подсчитать для сформированного массива произведение элементов главной диагонали (оформить в виде функции).
3. Составить блок-схемы

Вариант 7

1. Сформировать многомерный массив T размерностью 4x2 с помощью генератора случайных чисел в интервале от 0 до 100 (оформить в виде процедуры)
2. Посчитать для сформированного массива T сумму элементов первой строки (оформить в виде функции)
3. Составить блок-схемы

Вариант 8

1. Сформировать одномерный массив D, состоящий из 10 элементов с помощью генератора случайных чисел в интервале от 0 до 200 (оформить в виде процедуры)
2. Подсчитать для сформированного массива D среднее арифметическое (оформить в виде функции)
3. Составить блок-схемы

Вариант 9

Дана целочисленная квадратная матрица 4x4. Определить:

1) произведение положительных элементов матрицы (оформить в виде функции)
2) минимальный элемент во второй строке (оформить в виде процедуры)
3) составить блок-схемы

Вариант 10

Дана целочисленная прямоугольная матрица. Определить:

1) сумму отрицательных элементов (оформить в виде функции)
2) максимальный элемент в первой строке
3) составить блок-схемы

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое подпрограмма?
2. Что такое процедура?
3. Что такое функция?
4. Что такое фактический параметр?
5. Что такое формальный параметр?
6. Как фактические параметры должны соответствовать формальным параметрам?
7. Чем описание процедуры отличается от описания функции?
8. Как осуществляется вызов процедуры?
9. Как осуществляется вызов функции?

ПРАКТИЧЕСКАЯ РАБОТА 12

ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ТЕКСТОВЫХ ФАЙЛОВ

Цель работы: научиться объявлять текстовые файлы в программе, научиться создавать и обрабатывать текстовые файлы с помощью языка программирования Паскаль.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ ДАННЫЕ

В Паскале понятие **файла** употребляется в двух смыслах:

1. как поименованная информация на внешнем носителе (внешний файл);
2. как переменная файлового типа (внутренний файл)

Файловый тип переменной — это структурированный тип, представляющий собой совокупность однотипных элементов

Файл можно представить как последовательную цепочку элементов, пронумерованных от 0, заканчивающуюся специальным кодом, который называется **маркер конца**

Эл. 0	Эл. 1	….	Эл. n	Маркер конца

Структура описания файловой переменной

Var <имя переменной>: **File of** <тип элемента>
Например:

```
Var
    F1: file of Integer;
    F2: file of Real;
    F3: file of Char;
```

Файловая переменная связывается с именем файла в результате обращения к стандартной процедуре ASSIGN:

ASSIGN (<ф.п.>, <имя файла или л.у.>);

Здесь <ф.п.> - файловая переменная (правильный идентификатор, объявленный в программе как переменная файлового типа);
<имя файла> - текстовое выражение, содержащее имя файла

Инициализация файла
Инициировать файл означает указать для этого файла направление передачи данных. В Турбо Паскале можно открыть файл для чтения, для записи информации, а также для чтения и записи одновременно.

Процедура APPEND

инициирует запись в ранее существовавший текстовый файл для его расширения,
при этом указатель файла устанавливается в его конец. Процедура APPEND применима только к текстовым файлам, т.е. их файловая переменная должна иметь
тип TEXT . Процедурой APPEND нельзя инициировать запись в типизированный или нетипизированный файл. Если текстовый файл ранее уже был открыт с помощью RESET или

REWRITE, использование процедуры APPEND приведет к закрытию этого файла и открытию его вновь, но уже для добавления записей.

Процедура CLOSE

Закрывает файл, однако связь файловой переменной с именем файла,
установленная ранее процедурой ASSIGN, сохраняется. Формат обращения:
CLOSE (<ф.п.>)
При создании нового или расширении старого файла процедура обеспечивает сохранение в файле всех новых записей и регистрацию файла в каталоге. Функции процедуры CLOSE выполняются автоматически по отношению ко всем открытым файлам при нормальном завершении программы. Поскольку связь файла с файловой переменной сохраняется, файл можно повторно открыть без дополнительного использования процедуры ASSIGN.

Типы файлов

В Паскале различают текстовые, типизированные и нетипизированные файлы

Текстовые файлы предназначены для хранения текстовой информации. Для доступа к записям применяются процедуры READ, READLN, WRITE, WRITELN.
В этом случае осуществляется обращение к дисковому файлу или логическому устройству, связанному с переменной процедурой ASSIGN.

Приступая к решению задач этого раздела, следует вспомнить, что:

1. в программе, которая выводит результаты в файл или читает исходные данные из файла, должна быть объявлена файловая переменная типа text;
2. для доступа к конкретному файлу файловую переменную нужно связать с этим файлом (делается это при помощи инструкции assign);
3. для того, чтобы файл был доступен, его надо открыть (для ЧТЕНИЯ С ПОМОЩЬЮ ИНСТРУКЦИИ reset, ДЛЯ ЗАПИСИ — rewrite, для добавления — append);
4. при работе с файлами возможны ошибки, например, из-за того, что программа пытается открыть файл, которого нет, поэтому после каждой инструкции, которая может привести к возникновению ошибки, желательно, используя функцию iOResuit, проверять код завершения операции с файлом: чтобы программа могла контролировать результат выполнения операции с файлом, в ее текст надо поместить директиву
5. запись в файл выполняют инструкции write и writein, чтение — read и readin, причем в качестве первого параметра этих инструкций следует указывать файловую переменную;
6. по завершении работы с файлом его нужно обязательно закрыть инструкцией close;

файл, созданный программой, в которой тип файловой переменной объявлен как text, можно просмотреть при помощи редактора текста.

Пример 1. Написать программу, которая на сменном диске компьютера (A:) создает файл numbers.txt и записывает в него 5 введенных пользователем целых чисел.

```
{ Создает на диске A: файл и записывает в него5 целых чисел, введенных пользователем } var
f: text; { текстовый файл }
n: integer; { число }
i: integer; { счетчик чисел }
begin
writeln('Создание файла'); writeln('Введите пять целых чисел.1);
writeln('После ввода каждого числа нажимайте <Enter>');
```

```
Assign(f,'a:\numbers.txt');
Rewrite(f); { открыть в режиме перезаписи } for i:=l to 5 do begin
write('->');
readln(n);
writeln(f,n); end;
close(f); { закрыть файл }
 writeln('Введенные числа записаны в файл ','a:\numbers.txt');
readln;
end.
```

Пример 2. Написать программу, которая выводит на экран содержимое файла a:\numbers.txt.

```
{ Выводит на экран содержимое файла a:\numbers.txt } var
f: text; { текстовый файл } n: integer; { число } begin
writeln('Содержимое файла a:\nunibers.txt1);
writeln ('----------------------------') ;
Assign(f,'a:\numbers.txt'); Reset(f); { открыть файл для чтения } While not EOF(f) do { пока не достигнут конец файла } begin
readln(f,n); { прочитать число из файла } writeln(n); { вывести прочитанное число на экран } end;
Close(f); writeln ('-readln;
закрыть файл
end.
```

Пример 3. Вычисляет среднее арифметическое чисел, находящихся в файле a:\numbers.txt

```
var
f: text; { текстовый файл } n: integer; { число, прочитанное из файла } kol: integer; { кол-во прочитанных чисел } sum: integer; { сумма прочитанных чисел ) sa: real; { среднее арифметическое }
begin
writeln('Вычисление среднего арифметического чисел, writeln('находящихся в файле a:\numbers.txt'); writeln('Чтение из файла. Подождите.'); sum:=0; kol:-0;
Assign(f,'a:\numbers.txt'); Reset (f); { открыть файл для чтения } While not EOF(f) do { пока не достигнут конец begin
readln(f,n); { прочитать число из файла }
sum:=sum+n;
kol:=kol+l; end;
Close(f); { закрыть файл } sa:=sum/kol;
writeln('Прочитано чисел: ',kol); writeln('Сумма чисел: ',sum) ; writeln('Среднее арифметическое: ',sa:9:2); readln;
end.
```

ЗАДАНИЕ

1. Создать файл, состоящий из вещественных чисел. Определить количество нулевых значений в этом файле
2. Дан текстовый файл, содержащий произвольный текст. Определить чего в нем больше: русских букв или цифр.
3. Дан файл, содержащий текст на русском языке. Определить, входит ли заданное слово в указанный текст.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое файловая переменная?
2. Какова структура файла?
3. Как описывается файловая переменная в программе?
4. Что такое типизированный файл?
5. Что такое нетипизированный файл?
6. Что такое текстовый файл?
7. Как объявляется текстовый файл в программе?
8. Назовите алгоритм работы с текстовым файлом

ПРАКТИЧЕСКАЯ РАБОТА 13

ТЕМА: ОРГАНИЗАЦИЯ ПРОГРАММ С ИСПОЛЬЗОВАНИЕ ТИПИЗИРОВАННЫХ И НЕТИПИЗИРОВАННЫХ ФАЙЛОВ

Цель работы: научиться описывать в программе типизированные файлы, обрабатывать типизированные и нетипизированные файлы.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ ДАННЫЕ

Типизированные файлы

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (т.е. доступ к компоненту по его порядковому номеру).

Перед первым обращением к процедурам ввода-вывода указатель файла стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода-вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

Процедура READ

Обеспечивает чтение очередных компонентов типизированного файла.

Формат обращения: READ (<ф.п.>,<сп.ввода>)

Здесь <сп.ввода> - список ввода, содержащий одну или более переменных такого же типа, что и компоненты файла.

Файловая переменная <ф.п.> должна быть объявлена предложением FILE OF... и связана с именем файла процедурой ASSIGN. Файл необходимо открыть процедурой RESET. Если файл исчерпан, обращение к READ вызовет ошибку ввода-вывода.

Процедура WRITE

Используется для записи данных в типизированный файл.

Формат обращения: WRITE (<ф.п.>,<сп.вывода>)

Здесь <сп.вывода> - список вывода, содержащий одно или более выражений того же типа, что и компоненты файла.

Процедура SEEK

Смещает указатель файла к требуемому компоненту.

Формат обращения: SEEK (<ф.п.>,<N компонента>)

Здесь <N компонента> - выражение типа LONGINT, указывающее номер компонента файла. Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

Функция FILEPOS

Возвращает значение типа LONGINT, содержащее порядковый номер компонента файла, который будет обрабатываться следующей операцией ввода-вывода.

Формат обращения: FILEPOS (<ф.п.>)

Функцию нельзя использовать для текстовых файлов. Первый компонент файла имеет порядковый номер 0.

2. Нетипизированные файлы

Нетипизированные файлы объявляются как файловые переменные типа FILE и отличаются тем, что для них не указан тип компонентов. Отсутствие типа делает эти файлы, с одной стороны, совместимыми с любыми другими файлами, а с другой - позволяет организовать высокоскоростной обмен данными между диском и памятью.

Чтение файла осуществляется с помощью стандартной процедуры RESET:

RESET (<ф.п.>);

Здесь <ф.п.> - файловая переменная, связанная ранее процедурой ASSIGN с уже существующим файлом

При выполнении этой процедуры дисковый файл подготавливается к чтению информации. В результате специальная переменная-указатель, связанная с этим файлом, будет указывать на начало файла, т.е. на компонент с порядковым номером 0.

Стандартная процедура REWRITE (<ф.п.>).

инициирует запись информации в файл, связанный ранее с файловой переменной <ф.п.>. Процедурой REWRITE нельзя инициировать запись информации в ранее существовавший дисковый файл: при выполнении этой процедуры старый файл уничтожается и никаких сообщений об этом в программу не передается. Новый файл подготавливается к приему информации и его указатель принимает значение 0.

При инициации нетипизированного файла процедурами RESET или REWRITE можно указать длину записи нетипизированного файла в байтах.

Например, так:

```
var
f: file;
begin
.......
assign(f, 'myfile.dat')
reset(f,512);
.......
end.
```

Длина записи нетипизированного файла указывается вторым параметром при обращении к процедурам RESET или REWRITE, в качестве которого может использоваться выражение типа WORD. Если длина записи не указана, она принимается равной 128 байтам.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением READ и WRITE

Работа с файлам записей

Пример1. Сформировать файл Fm.dat, содержащий экзаменационную ведомость одной студенческой группы. Записи файла состоят из следующих полей: фамилия, имя, отчество, номер зачетной книжки, оценка.

```
Program examen;
Uses crt;
Type stud=record
    fio:string[30];
    Nz:string[6];
    mark:integer
    end;
Var Fstud:file of stud;
   S:stud;
   N,I:byte;
Begin
clrscr;
```

```
    Assign(Fstud, 'Fm.dat');
    Write('kolvo studentov');
    Readln(n);
    For I:=1 To n do
    Begin
        Write('vvedite fio:'); readln(s.fio);
        Write('vvedite nomer zachetki:'); readln(s.Nz);
        Write('vvedite ocenka:'); readln(s.mark);
        Write(Fstud,s)
    end;
    Writeln('Formirovanie zaversheno');
    Close(Fstud)
End.
```

Прямой доступ к записям файла

В стандарте языка Паскаль допустим только последовательный доступ к элементам файла. Одной из дополнительных возможностей, реализованный в Паскале, является прямой доступ к записям файла

Задав номер элемента файла, можно непосредственно установить указатель на него. После этого можно читать или перезаписывать данный элемент. Установка указателя на нужный элемент файла производится процедурой:

Процедура SEEK

Смещает указатель файла к требуемому компоненту. Формат обращения:

SEEK (<ф.п.>,<N компонента>)

Здесь <N компонента> - выражение типа LONGINT, указывающее номер компонента файла. Первый компонент файла имеет номер 0. Процедуру нельзя применять к текстовым файлам.

ЗАДАНИЕ

1. Записать в файл последовательного доступа N действительных чисел. Вычислить произведение компонентов файла.
2. В типизированном файле, компонентами которого являются целые числа, подсчитать количество отрицательных чисел.
3. В типизированном файле, компонентами которого являются вещественные числа, вычислить произведение положительных чисел.

ПРАКТИЧЕСКАЯ РАБОТА 14

ТЕМА: ПРОГРАММИРОВАНИЕ МОДУЛЯ

Цель работы: научиться применять в программах стандартные библиотеки подпрограмм, создавать собственные библиотеки подпрограмм.

Оборудование: ПК, ИСР Pascal ABC

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Стандартный Паскаль не располагает средствами разработки и поддержки библиотек программиста, которые компилируются отдельно и в дальнейшем могут быть использованы не только самим разработчиком. Если программист имеет достаточно большие наработки и если те или иные подпрограммы могут быть использованы при написании новых приложений, приходится эти подпрограммы целиком включать в новый текст.

В Паскале это ограничение преодолевается, во-первых, внедрением внешних подпрограмм, а, во-вторых, созданием и использованием модулей.

Внедрение внешних подпрограмм

В этом случае исходный текст каждой процедуры или функции хранится в отдельном файле и при необходимости с помощью специальной директивы компилятора включается в текст создаваемой программы.

Пример1. Дано число n, требуется найти сумму первой и последней цифр этого числа.

Создаем функцию:

```
Function Digits(N: LongInt): Byte;
Var Kol: Byte;
Begin
        Kol:=0;
        While N<>0 Do
        Begin
                Kol:=Kol + 1;
                N:=N Div 10
        End;
        Digits:=Kol
End;
```

Сохраним этот текст в файле с расширением .inc (расширение внешних подпрограмм), например digits.inc.

Опишем также функцию возведения числа в степень a^n:

```
Function Power (A,N: LongInt): LongInt; {файл power.inc}

Var I, St: LongInt;
Begin
        St:=1;
        For I:=1 To N Do
                St:=St*A;
```

```
        Power:=St
End;
```
Основная программа, решающая поставленную задачу с использованием описанных подпрограмм:

```
Program Example1;
Var N, S: Integer;
{$I digits.inc} {Подключение внешней функции из файла digits.inc, вычисляющей количество цифр в записи числа}
{$I  power.inc} {Подключение внешней функции из файла power.inc, вычисляющей результат возведения числа a в степень n }
Begin
        Write (‘Введите  целое число: ’);
        Readln(n); {Для определения последней цифры числа N берется остаток от деления этого числа на 10, а для определения первой цифры число N делится на число 10, возведенное в степень, на единицу меньшую, чем количество цифр в записи числа N}
        S:= N Mod 10 + N Div Power(10, Digits(N)-1);
        Writeln(‘Искомая сумма=’, S)
End.
```

Директива компилятора {$I <имя файла>} позволяет в данное место текста программы вставить содержимое файла с указанным именем.

Файлы с расширением .inc можно накапливать на жестком диске, формируя таким образом личную библиотеку подпрограмм.

Создание и использование модулей

Модуль – это набор ресурсов (функций, процедур, констант, переменных, типов и т.д.), разрабатываемых и хранимых независимо от использующих программ. В отличие от внешних подпрограмм, модуль может содержать достаточно большой набор процедур и функций, а также других ресурсов для разработки программ.

Стандартные модули:

- System
- Crt
- Graph и др.

Например, стандартный модуль CRT в паскале поддерживает 16 цветов, нумерация которых начинается с 0 (от 0 до 15 включительно). Также можно просто прописывать названия цветов не используя цифры.
Для того чтобы использовать модуль CRT его нужно подключить с помощью следующей строчки:

```
uses crt;
```

Для того чтобы закрасить фон нужно использовать такую команду:
textbackground (2); //Фон залит зелёным цветом
textbackground следует применять вместе с процедурой очистки экрана, которая описана ниже.
Для очистки экрана (1) и установки позиции курсора (2) используются соответственно следующие команды:
clrscr; {(1) – очистка курсора}
gotoxy (9,6); {(2) –Курсор установлен в точку (9;6)}
Цвет текста определяет процедура:
textcolor (13);
Для временной задуржки существует процедура delay
delay (1500);
Запускать программу в Pascal с использованием модуля CRT нужно при помощи клавиш Shift+F9.

Пример 2. Программа на Паскаль с использованием модуля CRT:

```
uses {Подключение модуля}
   crt;

begin
   textbackground (2); {Перекраска фона в зелёный}
   clrscr; {Очистка экрана и применение цвета фона}
   textcolor (13); {Текст будет розовым цветом}
   gotoxy (9,6); {Курсор будет перемещён в точку (9;6)}
   write ('Программирование'); {Вывод текста}
   delay (1500); {Задержка}
   textbackground (3); {Фон будет салатного цвета}
   clrscr; {Очистка экрана }
   gotoxy (20,20); {Курсор перемещён в точку (20;20)}
   textcolor (red); {Цвет текста – красный}
   write ('Алгоритмизация'); {Вывод текста}
   readln;
end.
```

Структура модуля:

Unit <имя модуля>; {Заголовок модуля}
Interface
{Интерфейсная часть}
Implementation
{Раздел реализации}
Begin
{Раздел инициализации модуля}
End.

Примечания:

- Имя модуля должно совпадать с именем файла и не должно содержать более 8 символов
- В разделе **Interface** объявляются все ресурсы, которые будут в дальнейшем доступны программисту при подключении модуля.
- В разделе **Implementation** описываются все подпрограммы, которые были ранее объявлены.
- Раздел **инициализации** содержит операторы, которые должны быть выполнены сразу же после запуска программы, использующей модуль. **Может быть пустым.**

При разработке модуля рекомендуется следующий порядок:

1. Спроектировать модуль, то есть определить основные и вспомогательные подпрограммы и другие ресурсы;
2. Описать компоненты модуля;
3. Отладить каждую программу отдельно, после чего «вклеить» в текст модуля
4. Откомпилировать модуль. Для этого в меню системы Паскаль выбрать **Compile/Destination Disk**, а затем – **Compile/Build**
5. Подключить разработанный модуль к программе, где планируется его использование. Для подключения модулей используется служебное слово **Uses**

Пример 3. Разработать модуль, содержащий подпрограмму суммирования элементов массива.

Разбиваем текст программы на две части: подпрограмму размещаем в модуле, а тестирующую программу оставляем в качестве основной программы. Так как все структурные типы параметров должны быть предварительно объявлены, описываем тип массива в модуле.

```
{Модуль должен размещаться в файле Summa.pas}
Unit Summa;

Interface {объявление внешних ресурсов}
Type mas=array[1..10] of integer;
Function sum(b:mas; n:integer):integer;

Implementation
Function sum; {описание функции}
Var
   s, i: integer;
begin
    s:=0;
    for i:=1 to n do s:=s+b[i];
    sum:=s;
end;

Begin

end.
```

Программа использует из модуля два ресурса: описание типа mas для объявления массива A и функцию Sum.

```
Program Ex;
Uses summa;  {указание используемого модуля}
Var
a: mas;   {используем ресурс mas}
i, n: integer;
s:real;
Begin
        Writeln('Введите размерность массива:');
        Readln(n);
        Writeln('Введите элементы массива:');
        For I:=1 To n do read(a[i]);
        s:=sum(a,n); {используем ресурс sum из модуля Summa}
        Writeln('summa=',s:5:2);
end.
```

Программа использует из модуля два ресурса: описание типа mas для объявления массива A и функцию Sum.

Задание.

1. Составить модуль по заданным подпрограммам (процедурам) для работы с телефонным справочником. Процедуры выполняют следующие действия:

1. Создание нового файла, который содержит данные: фамилию абонента, и его номер телефона;
2. Просмотр списка справочника;
3. Изменение записи справочника;
4. Поиск абонента по фамилии.

Доработать телефонный справочник, добавить в модуль и основную программу процедуру для поиска абонента по номеру телефона. Составить блок-схемы для основной программы и для любой подпрограммы.

1. Procedure Create_Book_Phone – процедура формирования телефонного справочника

```
Procedure Create_Book_Phone;
Var
  Ind, Count: integer;
Begin
   write('Введите имя файла телефонного справочника');
   Readln(Name);
   Assign(BookFile, Name);
   Rewrite(BookFile);  {Открыть новый файл для записи}
   Writeln('Создание записей файла',Name);
   Write ('Введите число записей в файле');
   Readln(Count);
   for Ind:=1 to Count do
                begin
                   writeln('Ввод записи №', FilePos(Bookfile)+1);
                   with work do
                          begin
                             write('Введите фамилию:  ');
                             readln(FIO);              {Ввод фамилии в файл}
                             write('  Введите номер телефона: ');
                             readln(Phone);            {Ввод номера телефона в файл}
                             Write(BookFile, Work);        {Записать в файл фамилию и номер
телефона}
                          end;
                end;
   Writeln('Создание файла справочника завершено');
   Writeln('Файл данных имеет ', FileSize(BookFile), 'записей');
   Close(BookFile);
end;
```

Пояснение: Для сокращения записи при обращении к полям FIO и Phone переменной типа запись Work используем предложение with work do. В окончании процедуры организуем вывод сообщения о результатах создания файла. Для измерения размера файла в записях используем функцию FileSize.

2. **Procedure OutputAllRec** – процедура просмотра списка телефонного справочника

```
Procedure OutputAllRec;
begin
Writeln('Введите имя файла');
Readln(Name);
Assign(BookFile, Name);
```

```
Reset(BookFile);
If IOResult = 0 then
        begin
            Seek(BookFile,0);
            Writeln('***Вывод телефонного справочника из файла ',  Name,  '***');
            While (not Eof(Bookfile)) do
                        begin
                            Read(BookFile, work);
                            With work do
                                    begin
                                        Write ('Zapis N', FilePos(BookFile), ':');
                                        Writeln('Фамилия Имя Отчество:', FIO, 'Телефон: ',
Phone);
                                    end;
                        end;
        end;
end;
```

Пояснение: для вывода на экран всех записей файла сначала, используя функцию **IOresult**, выполним проверку наличия данного файла на диске. Если функция **IOresult** возвращает значение, отличное от 0, то на экран выводится сообщение о том, что требуемого файла нет на диске. В противном случае разместить указатель так, чтобы он указывал на первую запись: **Seek(BookFile, 0)**, не забывая, что первый компонент имеет номер 0. Затем, используя оператор **while**, вызвать процедуру вывода на экран одной записи. Условием окончания цикла **while** будет выражение **Eof(BookFile)**. Как только оно выполнится, цикл завершится. Данные всех записей файла выводятся на экран. Для вывода на экран номера записи применим функцию **FilePos(BookFile).**

3. Procedure UpDateRec – процедура изменения записи телефонного справочника

```
Procedure UpDateRec;
var
  NumRec:LongInt;
begin
    Writeln('Введите имя файла');
    Readln(Name);
    Assign(Bookfile, Name);
    Reset(BookFile);
    If IOResult = 0 then
            begin
                Write('Укажите номер изменяемой записи: ');
                Readln(NumRec);
                Seek(BookFile, NumRec-1);          {Установка файловой позиции по
указанному номеру}
                Writeln('—старое значение записи:');
               {Вывод записи на экран и переход на следующую}
                Read(BookFile, work);
                With work do
                        begin
                            Write ('Запись №', FilePos(BookFile), ':');
                            Writeln('Фамилия Имя отчество:', FIO, '    Телефон: ', Phone);
                        end;
                Seek(BookFile, NumRec-1);          {Возврат на прежнюю позицию}
```

```
                Writeln('Задаем новое значение', NumRec, 'записи');
                begin
                   writeln('Вводим запись №', FilePos(Bookfile)+1);
                   with work do
                         begin
                            write('Введите фамилию:  ');
                            readln(FIO);
                            write('Введите номер телефон: ');
                            readln(Phone);
                            Write(BookFile, Work);
                         end;
                end;
                Close(BookFile);
             end;
end;
```

Пояснение: для изменения записи файла сначала, используя функцию **IOResult**, проверим, есть ли данный файл на диске. Дальше выводится запрос о номере изменяемой записи. После считывания номера записи переместим указатель к данной записи и выведем данные этой записи из файла на экран. Затем задаем новые значения полей этой записи. В разделе описания переменных опишем локальную переменную **NumRec** типа **LongInt** (длинное целое), которая будет принимать значение номера изменяемой записи.

4. Procedure FindFIO – процедура поиска абонента по фамилии

```
Procedure FindFIO;
var
  Bookfile: file of RecBook;
  Work: RecBook;
  Maska: StFIO;
  Rez_Find: boolean;
  CountRec: integer;
Begin
   Writeln('Укажите имя файла');
   Readln(name);
   Assign(BookFile, Name);
   Reset(BookFile);
   if IOresult = 0 then
          begin
            write('Введите фамилию для поиска:');
            Readln(Maska);
            Rez_Find:=False;
            CountRec:=0;
            While (not Eof(BookFile)) do
                 begin
                   Read(BookFile, Work);
                   With work do
                   if Pos(Maska, FIO)<>0 then
                     begin
                       Rez_Find:=True;
                       Inc(CountRec);
                       Writeln('Фамилия:' , FIO, 'Телефон:', Phone);
```

```
                    end;
                end;
            if Rez_Find then
            Writeln('Число записей для ', Maska, ' = ', CountRec)
                else
                    writeln('Нет абонентов с фамилией ', Maska);
            Close(BookFile);
        end;
```

Пояснение: в процедуре поиска номера телефона опишем локальные переменные: **Maska** типа **stFio**, которая будет хранить фамилию искомого абонента; **Rez_Find** типа Boolean, которая будет принимать значения True (истина) и False (ложь) в зависимости от результата поиска; **CountRec** , значение которой будет равно числу записей с такой фамилией.

После запроса имени файла данных телефонного справочника и проверки его наличия на диске сделаем запрос фамилии искомого абонента. Перед поиском присвоим переменным **Rez_Find** и **CountRec** значения false и 0 соответственно.

Просмотр всех записей файла данных при поиске реализуем при помощи оператора цикла **while**. Условием завершения поиска является **Eof(BookFile)** – достяжение конца файла. Для сокращения обращения к именам полей записи используем предложение **with Work do**.При совпадении значения поля FIO очередной записи со значением переменной Maska переменной Rez_Find присваивается значение True (абонент с заданной фамилией найден), значение переменной **CountRec** увеличивается на 1 и выводится сообщение о фамилии и номере телефона найденного абонента.

Описываем глобальные переменные, которые будут нужны в процессе работы процедур (Блок описания модуля)

```
type
   StFio = string[20];
   StPhone = string[10];
   RecBook = record          {запись сведений об абоненте}
            FIO: StFio;      {поле фамилия}
            Phone: StPhone   {поле номер телефона}
           end;
var
  BookFile: file of RecBook;   {Описание файловой переменной, компонентами, которой явл. запись}
  Work: RecBook;               {Переменная для доступа к записям}
   Name: string;                    {переменная, которая содержит имя внешнего файла на диске}
```

Основная программа, использующая ресурсы модуля telephon

```
Program Tel_spravochnik;
uses crt, telephon;
var
    Vid:byte;          {переменная, которая позволяет выбрать вид операции работы со справочником}
    End_Menu: boolean;

Begin
clrscr;
End_Menu:=False;
Repeat
```

```
    writeln('***Телефонный справочник***');
    Writeln('Выберите вид работ с телефонным справочником:');
    Writeln('1 – создание нового файла');
    Writeln('2 – просмотр списка справочника);
    Writeln('3 – изменение справочника');
    Writeln('4 – поиск абонента по фамилии');
    writeln('0 – завершение работы') ;
    Writeln('Ваш выбр:');
    Readln(Vid);
    Case vid of
        1: Create_Book_Phone;
        2: OutputAllRec;
        3: UpDateRec;
        4: FindFIO;
        0: End_Menu:=True;
    end;
    Writeln('Для продолжения Enter');
    Readln;
    Clrscr;
until End_Menu=true;
end.
```

ЗАДАНИЕ

Реализовать в виде модуля набор подпрограмм для выполнения следующих операций:

1. перевод из кг в граммы
2. перевод из кг в тонны
3. перевод из кг в центнер
4. перевод из кг в миллиграммы

Создать программу, в которой необходимо будет обратиться к ресурсам разработанного модуля.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Когда необходимо использовать модуль?
2. Чем модуль отличается от подпрограмм?
3. Из каких разделов состоит модуль?
4. Дайте характеристику разделов модуля?

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Основные источники:

1. Семакин И.Г. Основы программирования и баз данных : учебник для студ. учреждений сред. проф. образования / И.Г.Семакин М. : Издательский центр «Академия», 2014.
2. Семакин, И. Г. Основы алгоритмизации и программирования [Текст] : учебник для среднего профессионального образования/ И .Г. Семакин, А.П. Шестаков — 3-е изд., стер. — М. : Изд-во «Академия», 2012.

Дополнительные источники:

1. Семакин И.Г. Основы алгоритмизации и программирования. Практикум: учеб. пособие для студ. учреждений сред. проф. Образования / И.Г. семакин, А.П. Шестаков. — М. Издательский центр «Академия», 2013.
2. Фаронов В.В. Delphi 2005. Разработка приложений для баз данных и Интернета. — СПб.: Питер, 2006
3. Немнюгин С.А. Turbo Pascal. Практикум. 2-е изд. / СПб.: Питер, 2007
4. Немнюгин С.А. Turbo Pascal. Программирование на языке высокого уровня: Учебник для вузов. 2-е изд. — СПб.: 2006
5. Попов, В. Б. Паскаль и Дельфи. Самоучитель [Текст] — СПб. : Питер, 2005.