

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени А. А. КУЛЕШОВА»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

РАЗДЕЛ PASCAL

Учебно-методический комплекс

Составитель С. С. Новашинская



Могилев
МГУ имени А. А. Кулешова
2019

УДК 004.4(075.3)

ББК 32.97я72

О-75

*Печатается по решению редакционно-издательского совета
МГУ имени А. А. Кулешова*

Рецензенты:

доцент кафедры математики и информатики
МГУ имени А. А. Кулешов

Н. В. Кожуренко;

старший преподаватель кафедры
автоматизированных систем управления
Белорусско-Российского университета

Н. В. Выговская

Основы алгоритмизации и программирования: раздел Pascal :

О-75 учебно-методический комплекс / составитель С. С. Новашинская. – Могилев : МГУ имени А. А. Кулешова, 2019. – 112 с. : ил.

ISBN 978-985-568-505-1

Учебно-методический комплекс «Основы алгоритмизации и программирования: раздел Pascal» направлен на развитие у учащихся алгоритмического мышления, формирование знаний основных сведений о языке программирования Pascal и его конструкциях; формирование навыков разработки программ с использованием языка программирования Pascal; осуществление контроля знаний для проверки усвоения материала, выявления пробелов в знаниях учащихся, систематизацию знаний учащихся по мере прохождения темы.

Предназначен для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий». Комплекс может быть полезным для учащихся и учителей информатики общеобразовательных учреждений.

УДК 004.4(075.3)

ББК 32.97я72

ISBN 978-985-568-505-1

© Новашинская С. С., составление, 2019

© МГУ имени А. А. Кулешова, 2019

ОГЛАВЛЕНИЕ

Введение	5
Глава I. Теоретический раздел.....	6
Тема 1. Понятие алгоритма: свойства, способы описания	6
Тема 2. Структура программного модуля. Состав интегрированной программной среды	13
Тема 3. Условный оператор, оператор выбора. Логические операции в Pascal	22
Тема 4. Операторы организации циклов.....	26
Тема 5. Массивы: определение, описание, размещение в памяти, использование.....	32
Тема 6. Процедуры и функции.....	37
Тема 7. Символьные переменные и строки	40
Тема 8. Организация библиотек. Стандартные библиотечные модули и модули пользователя. Структура Unit-а	44
Глава II. Лабораторный практикум	50
Лабораторная работа № 1	51
Лабораторная работа № 2.....	53
Лабораторная работа № 3.....	55
Лабораторная работа № 4.....	57
Лабораторная работа № 5.....	59
Лабораторная работа № 6.....	62
Лабораторная работа № 7.....	63
Лабораторная работа № 8.....	64
Лабораторная работа № 9.....	66
Глава III. Контроль знаний	68
Тема 1: Понятие алгоритма: свойства, способы описания	68
Тест 1.1	68
Тест 1.2	69
Тема 2: Структура программного модуля. Состав интегрированной программной среды	71
Тест 2.1	71
Тест 2.2	72

Тест 2.3	73
Тест 2.4	75
Тест 2.5	76
Задание 2.1	77
Тема 3: Условный оператор, оператор выбора. Логические операции в pascal	79
Тест 3.1	79
Тест 3.2	81
Задание 3.1	83
Тема 4: Операторы организации циклов	83
Тест 4.1	83
Тест 4.2	85
Тест 4.3	87
Задание 4.1	90
Тема 5: Массивы: определение, описание, размещение в памяти, использование.....	91
Тест 5.1	91
Тест 5.2	92
Тест 5.3	94
Тест 5.4	96
Тема 6: Процедуры и функции	99
Тема 7. Символьные переменные и строки	101
Тест 7.1	101
Тест 7.2	102
Тема 8. Организация библиотек. Стандартные библиотечные модули и модули пользователя. Структура Unit-а.....	104
Глава IV. Вспомогательный раздел.....	106
Зачетные задания	106
Перечень литературы по дисциплине «Основы алгоритмизации и программирования».....	107
Заключение	110
Список использованных источников	111

ВВЕДЕНИЕ

Основной целью изучения дисциплины «Основы алгоритмизации и программирования» является формирование профессиональной компетентности будущих специалистов в области принципов построения алгоритмов, структурного проектирования и методов разработки программ.

Программой дисциплины определены цели по каждой теме и спрогнозированы результаты их достижения в соответствии с уровнями усвоения учебного материала.

Задачей дисциплины является:

- развитие у учащихся алгоритмического мышления;
- формирование знаний о свойствах алгоритмов;
- приобретение практических навыков разработки программ с использованием языков программирования.

Дисциплина «Основы алгоритмизации и программирования» тесно связана с областями знаний ряда дисциплин: «Информатика», «Математика», «Операционные системы» и др. Знания, умения и навыки, полученные учащимися при изучении дисциплины «Основы алгоритмизации и программирования», являются основой изучения дисциплин «Конструирование программ и языки программирования», «Системное программирование», «Технология разработки программного обеспечения», «Базы данных и системы управления базами данных».

Разработанный учебно-методический комплекс «Основы алгоритмизации и программирования: раздел Pascal» содержит:

- основные сведения о языке программирования Pascal и его конструкциях;
- ряд лабораторных работ, в результате выполнения которых учащиеся приобретут необходимые знания и практические навыки программирования;
- контроль знаний для проверки усвоения материала, выявления пробелов в знаниях учащихся, систематизации знаний учащихся по мере прохождения новой темы;
- зачетные задания;
- список литературы по дисциплине «Основы алгоритмизации и программирования».

Содержание учебно-методического комплекса «Основы алгоритмизации и программирования: раздел Pascal» соответствует программе дисциплины «Основы алгоритмизации и программирования» для подготовки техников-программистов специальности 2-40 01 01 «Программное обеспечение информационных технологий».

Глава I. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

Тема 1. Понятие алгоритма: свойства, способы описания

Фундаментальным понятием при программировании является алгоритм. Написанию программы предшествует разработка алгоритма решения задачи.

В информатике понятие алгоритма является одним из основных [6]. Слово «алгоритм» происходит от имени персидского математика аль-Хорезми, что означает «из Хорезма», который в IX в. Разработал правила арифметических действий над десятичными числами. Примерами таких алгоритмов могут служить описания способов решения квадратных уравнений, нахождения наибольшего общего делителя и вычисления сложных процентов.

Алгоритм – это четкое описание последовательности действий, приводящих к решению задачи. Перечень предписаний, которые понимает и может выполнить исполнитель, называют его *системой команд*. Для составления алгоритма необходимо знать систему команд исполнителя, правила их записи и всего алгоритма в целом.

Алгоритм обладает перечисленными ниже *свойствами (особенностями)* [6]:

Дискретность (от лат. *discretus* – разделенный, прерывистый) указывает, что любой алгоритм должен состоять из конкретных действий, следующих в определенном порядке. Структура алгоритма оказывается дискретной: только выполнив одну команду, исполнитель сможет приступить к выполнению следующей.

Детерминированность (от лат. *determinate* – определенность, точность) указывает, что любое действие алгоритма должно быть строго и недвусмысленно определено в каждом случае. Многократное применение одного алгоритма к одному и тому же набору исходных данных должно всегда давать один и тот же результат.

Конечность определяет, что каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения.

Результативность требует, чтобы в алгоритме не было ошибок, т. е. при точном исполнении всех команд процесс решения задачи должен быть получен результат, определенный постановкой задачи.

Массовость показывает, что алгоритм можно использовать с разными исходными данными, т.е. применять при решении всего класса задач данного типа, отвечающих общей постановке задачи.

Формальность показывает, что алгоритм не должен допускать неоднозначность толкования действий для исполнителя.

Способы записи алгоритмов.

Наиболее часто используются следующие способы записи алгоритмов [6]: словесная, псевдокоды, алгоритмические языки, графические формы.

Если алгоритм предназначен для человека, то в качестве команд можно использовать слова, фразы и предложения. Такая форма записи алгоритма называется *словесной*. Простейшими примерами словесной формы записи алгоритмов, часто встречающимися в повседневной практики, являются инструкции по использованию бытовых радио- и электроприборов, телефонов-автоматов и т. п.

Словесная форма достаточно удобна для записи небольших алгоритмов. Однако если алгоритм описывает процесс, состоящий из десятков команд, то в такой форме он становится труднообозримым. Поэтому на практике обычно используют формализованные словесные формы записи или *псевдокоды* (наиболее известным примером псевдокода является учебный алгоритмический язык Кумир), *алгоритмические языки*, например, Паскаль, и *графические формы записи алгоритмов*, например, *схемы алгоритмов*.

В зависимости от порядка исполнения команд можно выделить три типа алгоритмов: *линейные*, *разветвляющиеся* и с повторениями (*циклические* алгоритмы).

Алгоритмы решения задачи называется *линейным*, если все его команды исполняются одна за другой, в порядке их записи.

Алгоритм называется *разветвляющимся*, если после проверки условия в различных ситуациях исполняются разные наборы команд.

При составлении алгоритмов решения задач одну и ту же команду часто приходится записывать несколько раз подряд. В этих случаях сокращения длины алгоритма используются специальные команды повторения. Алгоритмы, которые содержат команду повторения, называются *алгоритмы с повторениями* (*циклическими алгоритмами*).

В информатике исходные данные и результаты (промежуточные и итоговые) называют *величинами*. Обычно величины обозначают буквами, словами, сокращениями слов. Эти обозначения называют *именами величин*, например: a, sum, max. В именах величин можно использовать и цифры, но первой всегда должна быть буква, например: x1, d6b5, sena.

При записи арифметических выражений используются круглые скобки, знаки операций, числа, величины, имена математических функций, аргументы которых тоже записываются в круглых скобках, например: $-5*(a+\sin(a))$.

Основными операциями являются сложение, умножение, вычитание, деление и возведение в степень.

Операции в выражениях выполняются по обычным правилам. Сначала вычисляются значения функции, затем выполняется возведение в степень, далее умножение, деление и, наконец, сложение и вычитание. При необходимости порядок исполнения можно изменить с помощью скобок.

Значения величин задаются специальной командой, которую называют *командой присваивания*. В общем виде она выглядит так:

Имя величины := выражение.

Знак «:=» называется *знаком присваивания*. Порядок выполнения команды присваивания следующий [6]: сначала вычисляется значение выражения, за-

писано в правой части команды. Затем величине, имя которой указано слева, присваивается вычисленное значение. Поэтому команда присваивания читается так: присвоить значение выражения «выражение» величине «имя величины». Например, команда $sum := sum + b$. Читается так: «присвоить значение выражения $sum+b$ величине sum ».

При исполнении алгоритма значения всех используемых величин хранятся в памяти компьютера. Поскольку целые числа, десятичные дроби, тексты записываются в памяти по-разному, компьютеру нужно указывать, какого типа значения будет принимать та или иная величина. Обычно используют величины следующих типов: целый, вещественный, литерный. Для их обозначения применяются служебные слова, например, в алгоритмическом языке Pascal это *integer, real, char* соответственно.

Величины *целого* типа могут принимать в качестве значения только целые числа. Если значения величины – десятичные дроби, то тип величины *вещественный*, и, наконец, если в качестве значений величины необходимо использовать буквы, слова, предложения, то такая величина имеет *литерный тип*.

С понятием типа в информатике связан не только способ хранения значений величин в памяти компьютера, но и диапазон допустимых значений данных величин, а также набор допустимых операций над величинами данного типа (например, над данными целого тип допустима операция вычисления остатка от деления одного целого числа на другое, которая не имеет смысла для данных других типов).

В тех случаях, когда алгоритм содержит большое число ветвлений и циклов, его запись на псевдокоде не очень наглядна. Для того чтобы сделать процесс составления и анализа алгоритма более удобным для восприятия, используют *схему алгоритмов* – специально созданную графическую форму записи.

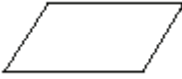

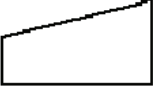
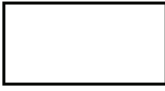

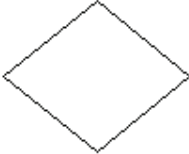
Схема алгоритма – это графическое представление алгоритма, дополненное элементами словесной записи. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой – *блоком (блочным символом)*, причем различным по типу выполняемых действий блокам соответствуют разные геометрические фигуры. Правила выполнения схем алгоритмов, применяемые графические символы, отражающие основные операции процесса обработки данных, регламентирует ГОСТ 19.701-90.

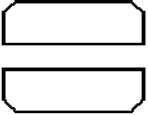
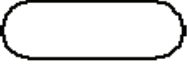
Блок-схема алгоритма – представление алгоритма в графической форме, в которой действия над данными изображаются в виде геометрических блоков с поясняющими надписями, а последовательность действий указывается соединительными линиями. Эта форма используется для наглядного представления процессов обработки данных, особенно в случаях с большим количеством проверок логических условий и разветвлений процессов (выбора последующих действий в зависимости от результатов предыдущих действий).

Основные графические символы (блоки) [1] представлены в таблице 1.1.1.

Таблица 1.1.1.

Символ	Наименование	Действия
--------	--------------	----------

	Данные	Символ отражает данные, носитель не определен.
	Документ	Символ отображает данные, представленные на носителе в удобно читаемой форме (машинограмма, документ для оптического или магнитного считывания, микрофильм, рулон ленты с итоговыми данными, бланки ввода данных).
	Ручной ввод	Символ отображает данные, вводимые вручную во время обработки с устройств любого типа (клавиатура, переключатели).
	Процесс	Символ отображает функции обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации).
	Предопределенный процесс	Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).
	Решение	Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

	Границы цикла	Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.
	Терминатор	Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).

Графические символы на схемах соединяются *линиями потока информации*. Основное направление потока информации идет сверху вниз и слева направо (здесь стрелки на линиях можно не указывать!). В других случаях применение стрелок обязательно. На рисунке приведены примеры некоторых условий графических обозначений, используемых на схемах.

Программа линейной структуры – программа, реализующая алгоритм структуры «Следование» (рис. 1.1.1), в котором блоки выполняются в указанном порядке, последовательно друг за другом.

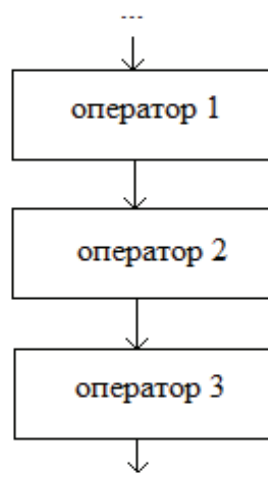


Рисунок 1.1.1 – Структура «Следование»

Программа разветвляющейся структуры – программа, реализующая алгоритм структуры «Ветвление» (рис. 1.1.2), который предусматривает выбор одной из нескольких последовательностей операторов (ветвей) в зависимости от некоторых условий.

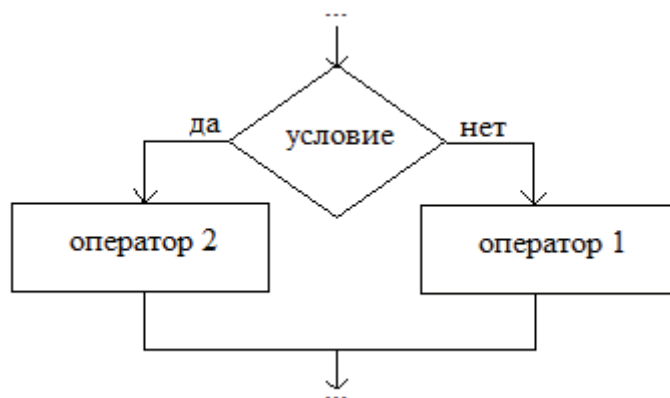


Рисунок 1.1.2 – Структура «Ветвление»

Программа циклической структуры – программа, реализующая алгоритм структуры «Цикл» (рис. 1.1.3-1.1.5), в котором происходит многократное повторение одного и того же участка программы. Различают детерминированные циклы с заранее известным числом повторений (цикл с параметром) и итерационные циклы (циклы с предусловием и с постусловием), в которых число повторений заранее неизвестно.

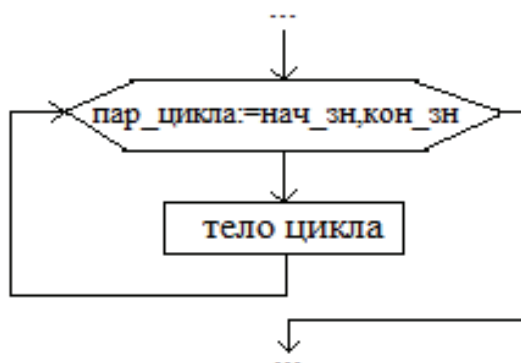


Рисунок 1.1.3 – Структура «Цикл «Для»

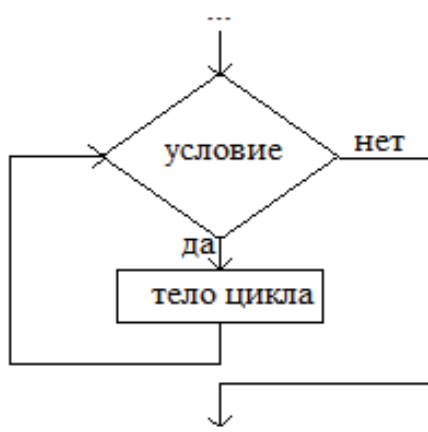


Рисунок 1.1.4 – Структура «Цикл с предусловием»

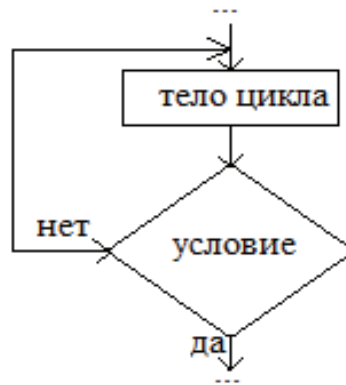


Рисунок 1.1.5 – Структура «Цикл с постусловием»

По отношению к блоку линии потока могут быть *входящими* или *выходящими*. Количество входящих линий для блока принципиально не ограничено. Выходящая линия может быть только одна. Исключение составляют логические блоки, имеющие не менее двух выходящих линий потока, каждая из которых соответствует одному из возможных исходов проверки логического условия, а также блоки модификации.

Внутри блоков и рядом с ними проставляют записи и обозначения (для уточнения выполняемых ими функций) таким образом, чтобы их можно было читать слева направо и сверху вниз независимо от направления потока.

Схема алгоритма является достаточно наглядным и простым способом записи алгоритма, если она занимает одну страницу А4. Схемы сложных больших алгоритмов обычно не помещаются на одном листе, поэтому целесообразно составлять несколько схем. Одна из них – основная, отображает общую идею алгоритма без излишних подробностей. Другие раскрывают содержание отдельных блоков основной схемы. При необходимости составляют схемы последующих уровней с еще большей степенью детализации.

Контрольные вопросы и задания:

1. Что такое алгоритм?
2. Назовите основные свойства алгоритмов.
3. Какие способы записи алгоритмов используют наиболее часто? Почему?
4. Что такое псевдокод?
5. Какой алгоритм называется линейным?
6. Какой алгоритм называется разветвляющимся?
7. Какой алгоритм называется циклическим?

Тема 2. Структура программного модуля. Состав интегрированной программной среды

Общие сведения.

Паскаль (англ. *Pascal*) – строго типизированный язык программирования общего назначения. Один из наиболее известных языков программирования, содержащий основные конструкции и способы обработки данных, присущие всем современным языкам программирования. Он является базой для таких языков, как Object Pascal, C++, Java, C# и др. Язык Паскаль был создан Никлаусом Виртом в 1968–1969 гг. после его участия в работе комитета разработки стандарта языка Алгол-68. Он был опубликован в 1970 г. Виртом как небольшой и эффективный язык, способствующий хорошему стилю программирования, использующий структурное программирование и структурированные данные. Язык Паскаль поддерживает технологии объектно-ориентированного программирования.

Наименьшая единица трансляции на языке Pascal выглядит следующим образом:

```
program map;  
begin  
end.
```

Этот код не выполняет никаких действий, но представляет собой законченную программу и содержит шесть лексем.

Лексема – это минимальная значимая (смыслообразующая) единица языка (текста программы), т. е. символ или последовательность символов, имеющая самостоятельное значение. В данном случае лексемами являются три ключевых слова `program`, `begin`, `end`, идентификатор (имя программы `map`) и символы-разделители (точка и точка с запятой) (иногда пробел, табуляцию и перевод строки также считают лексемами).

Алфавитом языка программирования Pascal называют совокупность всех допустимых символов, которые можно использовать в этом языке.

Алфавит Pascal включает в себя следующие символы:

- прописные и строчные буквы латинского алфавита от A до Z, а также символ подчеркивания (`_`);
- арабские цифры от 0 до 9;
- специальные одиночные знаки: `+ - * / = < > . , ; ^ $ # @`;
- специальные парные знаки: `[] , () , {}`;
- составные знаки: `<= , >= , < > , .. , (* *), (.)`;
- буквы русского алфавита, но они могут использоваться только при вводе и выводе данных строкового типа (т.е. при вводе и выводе текста, заключенного в апострофы (`' '`), или в комментариях к программе).

Синтаксис – это правила записи слов, выражений, операторов и других составных частей (языковых конструкций) текста на данном языке.

Структура программы.

В общем виде программа на языке Pascal включает в себя следующие части:

- заголовок программы;
- список подключаемых модулей;
- объявление типов, констант, переменных;
- объявление процедур и функций;
- основной программный блок;
- определение процедур и функций;
- комментарии.

В рамках синтаксиса языка общий вид программы:

```
program {имя программы};
uses {список подключаемых модулей через запятую}
label {блок описания меток}
const {блок объявления глобальных констант}
type {блок объявления пользовательских типов}
var {блок объявления глобальных переменных}
function {объявление функций}
{локальные переменные}
begin
...{операторы, составляющие тело функции}
end;
procedure {объявление процедуры}
{локальные переменные}
begin
...{операторы, составляющие тело процедуры}
end;

begin {начало основного блока программы}
...{исполняемые операторы}
end. {конец основного блока программы}
```

Идентификаторы, константы, метки.

Имена переменных, констант, меток, типов, модулей, процедур и функций, используемых в программе, называются *идентификаторами*. Идентификатор должен начинаться с буквы и может содержать буквы латинского алфавита, цифры и знаки препинания. Символ подчеркивания «_» также считается буквой. Исключения составляют метки, которые могут начинаться также и с цифры. Идентификатор может состоять из букв, цифр и знаков подчеркивания (пробелы, точки и другие специальные символы недопустимы). Между двумя идентификаторами должен быть, по крайней мере, один пробел. Максимальная длина идентификатора составляет 127 символов, но значимыми являются только первые 63 символа. При записи идентификаторов можно использовать как прописные, так и строчные буквы.

Желательно выбирать мнемонические имена, т.е. несущие смысловую нагрузку, как, например, result, summa, sena. Использование осмысленных имен предпочтительнее, т.к. это делает программу более простой для понимания.

Например: a1, b_2, k123, _d – идентификаторы; 1a, n2, @ru – не идентификаторы;

Служебные слова (ключевые) являются зарезервированными и не могут быть использованы в качестве идентификаторов. Примерами ключевых слов могут выступать: break, begin, program, else, to, until и т. д.

Постоянной (константой) называется величина, значение которой не изменяется в процессе исполнения алгоритма. Раздел описания именованных констант начинается со служебного слова const, после которого следуют строки вида: имя константы=значение; или имя константы: тип = значение;

Например:

```
const
Number=10;
Name='Victor' ;
```

Меткой можно именовать оператор. Метка строится аналогично общему правилу построения идентификаторов (символьная) или представляет собой последовательность арабских цифр (числовая). Метки используются для безусловного перехода с помощью оператора goto. Метка отделяется от оператора двоеточием.

Например,

```
Label 10, 20, 30, 40, 50, 60, 70;
Var x : real;
Begin
x:=4;          goto 50;
10: x:=sqr(x);   goto 40;
20: writeln(x);  goto 70;
30: x:=x-1;      goto 10;
40: x:=x*2+2;    goto 60;
50: x:=x+x;      goto 30;
60: x:=sqrt(x);  goto 20;
70: end.
```

Переменные, комментарии и типы данных.

Переменной называется величина, значение которой меняется в процессе исполнения алгоритма. Переменные – это некоторые данные, обрабатываемые в программе и имеющие имя. Имя переменной считается неизменным до конца программы, а значение самой переменной может меняться.

Раздел описания переменных начинается со служебного слова var (сокр. от англ. *variable* – переменная), после которого следуют элементы описания. Переменные могут описываться как в начале программы, так и непосредственно внутри любого блока begin...end. Кроме того, переменные-параметры цикла могут описываться в заголовке оператора for (цикл с параметром). Имена переменных в списке перечисляются через запятую.

В зависимости от версии языка программирования Pascal типы переменных могут немного различаться. В программе на языке Pascal все данные (переменные, константы, массивы, метки, подпрограммы пользователя) должны быть обязательно объявлены.

Объявить – значит определить тип и присвоить имя для того, чтобы программа при запуске смогла зарезервировать в оперативной памяти компьютера столько места, сколько необходимо для хранения переменной, константы, массива, метки и пр.

Тип данных – фундаментальное понятие теории программирования. Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям, и способ реализации хранения значений и выполнения операций. Любые данные, которыми оперируют программы, относятся к определенным типам.

Каждый язык программирования поддерживает один или несколько *встроенных типов данных* (базовых типов), кроме того, многие языки программирования предоставляют программисту возможность описывать *собственные (пользовательские) типы данных*. Приведем некоторые часто используемые *стандартные типы данных* языка Pascal:

Таблица 1.2.1.

	Название типа	Идентификатор	Диапазон значений	Размер памяти
Целые типы	Короткое целое со знаком	Shortint	-128...127	1 байт
	Целое со знаком	Integer	-32768...32767	2 байта
	Длинное целое со знаком	Longint	-2147483648... 2147483647	4 байта
	Целое без знака	Word	0...65535	2 байта
	Короткое целое	Byte	0...255	1 байт
Вещественные типы	Вещественное	Real	от $2.9 \cdot 10^{-39}$ до $1.7 \cdot 10^{38}$	6 байт
	Вещественное двойной точности	Double	от $5.0 \cdot 10^{-324}$ до $1.7 \cdot 10^{308}$	8 байт
Булевские типы	Булевское	Boolean	True или false	1 байт
Символьные типы	Символьное	Char	1 символ таблицы ASCII	1 байт
Строковые типы	Строковое	String	Последовательность символов ASCII длиной до 255	На 1 символ 1 байт

Все простые типы (кроме вещественного) являются *порядковыми*. Значения только этих типов могут быть индексами переменных и массивов и параметрами цикла `for`.

Предложение описания переменных начинается с зарезервированного слова `var`, далее, через запятую, указываются имена используемых переменных и через символ «:» тип переменных.

Пример:

```
var
x, y: real;
i, j: integer;
s: string;
a: array[1..20] of integer;
b: 1..100;
```

Приведем некоторые часто используемые *пользовательские типы данных* языка Pascal:

– *перечисления* – описывают новые типы данных, значения которых определяет сам программист. Список элементов этого типа заключается в круглые скобки (например, `var Mesyac: (May, June, July, August);`);

– *интервальный тип* – диапазон значений какого-то порядкового типа, называемого базовым. Список допустимых значений этого типа задается через указание минимального и максимального значений из некоторого диапазона через две точки (`var a: 0..10; c: 'a'..'n'; d: Mon..Sat;`);

– *структурированный тип* служит для описания данных, которые могут иметь много значений: тип-массив, множество, запись, файловый тип.

Комментарии не принимаются во внимание компилятором при построении исполняемого файла, но являются важной частью программы. Комментарии могут (и должны) содержать осмысленные пояснительный текст об используемых данных и алгоритмах. С помощью комментариев можно временно выключать из компиляции участки кода.

Комментарии заключаются между скобками `{..}`, `(*..*)` или пишутся после символов `«//»` (например, `// это – комментарий; {это – комментарий}; (*это – комментарий*)`).

```
Begin
//Write ('Введите числители и знаменатели дробей: ');
Readln (a, b, c, d);
```

Комментарий в программе отображается зеленым цветом.

Выражения, операнды и операции.

Выражение – формальное правило вычисления нового значения. Выражения подразделяют на группы в зависимости от данных (переменных, констант, функций), входящих в эти выражения; например: арифметические, логические, строковые и др.

Простыми выражениями являются переменные и константы. *Сложные* (составные) выражения строятся из более простых с использованием операций, скобок, вызовов функций, процедур, индексов и приведений типов.

Данные, к которым применяются операции, называются *операндами*.

Операциями в языках программирования называются действия над данными (операндами).

В языке Pascal имеются следующие операции:

1. Арифметические (бинарные (+, -, *, /, div, mod) и унарные (+, -) к одному операнду).

При выполнении целочисленного деления (div) остаток от деления отбрасывается. Например:

$$12 \text{ div } 4 = 3; 19 \text{ div } 5 = 3.$$

С помощью операции mod можно найти остаток от деления одного целого числа на другое:

$$12 \text{ mod } 3 = 0; 19 \text{ mod } 5 = 4.$$

2. Операции отношения (сравнения):

= – равно;

<> – не равно;

> – больше;

< – меньше;

<= – меньше или равно;

>= – больше или равно.

В операциях сравнения должны участвовать операнды одного типа. Исключение сделано только в отношении данных числовых типов real и integer, которые могут сравниваться друг с другом.

3. Строковые операции: основной операцией (помимо операций отношений) является операция конкатенации («+», слияния, сцепление строк).

Например, 'a'+ 'b'='ab' ('пол' + 'нота'='полнота').

4. Операция @ применяется к переменной и возвращает ее адрес.

5. Операции инкремента и декремента:

Inc(A) – инкремент, увеличение значения A на единицу;

Dec(A) – декремент, уменьшение значения A на единицу;

Inc(A,n) – увеличение значения A на n;

Dec(A,n) – уменьшение значения A на n.

6. Логические операции:

not – логическое отрицание;

and – логическое умножение (и);

or – логическое сложение (или);

xor – логическое исключающее или.

Приоритет операций

1 приоритет (наивысший): @, not;

2 приоритет: *, /, div, mod, and;

3 приоритет: +, -, or, xor;

4 приоритет (низший): =, <>, <, >, <=, >=, in.

Порядок действий при вычислении значения выражения.

1) вычисляются значения в скобках;

2) вычисляются значения функций;

3) выполняются унарные операции;

4) выполняются операции умножения и деления (в том числе целочисленного деления и нахождения остатка от деления);

5) выполняются операции сложения и вычитания.

Стандартные функции.

Функция – в программировании – это поименованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо [1]. На вход функции поступают управляющие воздействия в виде значений аргументов. На выходе функция возвращает результат, который может быть как скалярной величиной, так и векторным значением (структура, индексный массив и т. п.).

Многие действия с числовыми данными выполняются путем вызова встроенных в языки программирования функций. Такие функции называются *стандартными* ($\text{sqrt}(x)$, $\text{abs}(x)$ и т. д.). Существуют и функции пользователя. Приведем стандартные функции языка Pascal, используемые при написании программ (полный список встроенных функций можно найти в справочной системе среды программирования):

Таблица 1.2.2

Имя функции	Действие
Abs(x)	возвращает абсолютное значение (модуль) x
Sqr(x)	возвращает квадрат x
Sqrt(x)	возвращает квадратный корень из x
Sin(x)	возвращает синус x
Cos(x)	возвращает косинус x
Ln(x)	возвращает натуральный логарифм x
Exp(x)	возвращает e в степени x ($e=2.718281\dots$)
Arctan(x)	возвращает арктангенс x
Power (x,y)	возвращает x в степени y
Round(x)	возвращает результат округления x до ближайшего целого
Trunc(x)	возвращает целую часть x
Int(x)	возвращает целую часть x
Frac(x)	возвращает дробную часть x
Random(x)	возвращает случайное целое число в диапазоне от 0 до x-1
Random	возвращает случайное вещественное число в диапазоне [0..1)

Ввод и вывод данных, операция присваивания.

Основная часть программы состоит из команд, которые обрабатывают данные. Одной из таких команд является *команда присваивания*. С помощью команды присваивания можно задавать значения переменных (вводить исходные данные), вычислять значения арифметических выражений и результат записывать в переменные.

В команде присваивания слева от «:=» пишется имя переменной, а справа может быть число, переменная либо выражение.

Пример:

$c := (a+b) * 2; b := 5; a := b$

Рассмотрим схему, по которой происходит исполнение программы на языке Pascal:

- 1) *ввод* исходных данных с клавиатуры, из файла или с носителя информации;

- 2) *обработка* данных с помощью операторов языка Pascal;
- 3) *вывод* результатов обработки на экран, принтер, в файл или на носитель информации.

Для того чтобы ввести или вывести данные, необходимо выполнить определенные команды (процедуры). Процедура, которая в режиме диалога с клавиатуры присваивает значение для переменной величины, называется *процедурой ввода*.

В Pascal эта команда выглядит следующим образом: *read* (*список переменных*).

Синтаксис:

read (x_1, x_2, \dots, x_n); – осуществляет ввод x_1, x_2, \dots, x_n с клавиатуры;
readln (x_1, x_2, \dots, x_n); – осуществляет ввод x_1, x_2, \dots, x_n с клавиатуры и после выбора значения последней переменной обеспечивает переход к началу новой строки;
readln; – обеспечивает пропуск одной строки и переход к началу новой строки.

Элементами списка ввода могут быть имена переменных, которые должны быть заполнены значениями, введенными с клавиатуры. Выполнение операторов ввода происходит так: ход программы приостанавливается, на экран выводится курсор, пользователь с клавиатуры вводит необходимые значения в том порядке, в котором они требуются списком ввода, нажимает Enter (данные при вводе разделяются пробелами (в случае использования *read*)). После этого набранные данные попадают в соответствующие им переменные, и выполнение программы продолжается.

Процедура, которая выводит содержимое переменных на экран, называется *процедурой вывода на экран*.

В Pascal эта команда выглядит следующим образом: *write* (*список переменных*).

Синтаксис:

write (x_1, x_2, \dots, x_n); – выполняет вывод значений, соответствующих перечисленным именам, размещая выводимые значения в одной строке;
writeln (x_1, x_2, \dots, x_n); – выполняет вывод значений x_1, x_2, \dots, x_n , соответствующих перечисленным именам, и после вывода последнего значения осуществляет переход к новой строке;
writeln; – обеспечивает пропуск строки и переход к началу новой строки.

Элементами списка вывода могут являться имена переменных, выражения, константы. Прежде чем вывести на экран, компьютер вычислит значения выражений.

Формат вывода данных можно задать в скобках, указав после двоеточия (:) количество символов, выделяемых для выводимого значения переменных.

Задание формата выводимых данных:

– для вывода значений целого типа:

```
write (a:m);  
writeln (a:m);
```

– для вывода вещественного типа:

```
write (a:m:n);  
writeln (a:m:n);
```

где *a* – значение переменной выводимой переменной; *m* – ширина всего поля вывода; *n* – часть поля, отводимого под дробную часть числа.

Пример:

```
var x: integer;  
    y: real;  
Begin  
x:=9999;  
y:=25.89;  
writeln (x:7);  
writeln (y:7:2);  
end.
```

Результат вывода: __ _9999.
 __ _25.89

Контрольные вопросы и задания:

1. Что такое лексема, алфавит, синтаксис?
2. Какие символы образуют алфавит языка Pascal?
3. Определите общую структуру программы.
4. Для чего предназначены переменные, метки, константы, комментарии?
5. Перечислите стандартные типы данных в Pascal.
6. Укажите отличие данных вещественного и целого типов.
7. Назовите логические операции.
8. Что представляет собой условие?
9. Что понимается под символьными данными?
10. Какие данные называют арифметическими?
11. Что такое функция? Назовите основные стандартные функции языка.
12. Назовите операторы ввода, вывода.
13. Особенность формата вывода данных.

Тема 3. Условный оператор, оператор выбора. Логические операции в Pascal

Разветвляющие процессы в языке Pascal реализуются с помощью условного оператора `if` и оператора выбора `case`.

Оператор безусловного перехода `goto` вызывает передачу управления оператору, которому предшествует метка, указанная в данном операторе перехода [1].

Синтаксис оператора безусловного перехода:

```
label n;  
...  
goto n;  
...  
n: оператор;  
...
```

где `n` – метка оператора.

Пример:

```
Label 1, 2, 3, 4, 5, 6, 7;  
Var y: real;  
Begin  
y:=5;          goto 5;  
1: y := y/8;   goto 4;  
2: writeln(y); goto 7;  
3: y := y-1;   goto 1;  
4: y := y*5+1; goto 6;  
5: y := sqr(y); goto 3;  
6: y := sqrt(y); goto 2;  
7: end.
```

Использовать оператор `goto` можно и в случаях, например, для выхода к концу программы или процедуры в случае неправильного задания данных или выхода из тела цикла.

Условный оператор.

Выбор из двух возможностей реализуется *условным оператором*. Графическая схема условного оператора `if` представлена на рис.1.3.1.

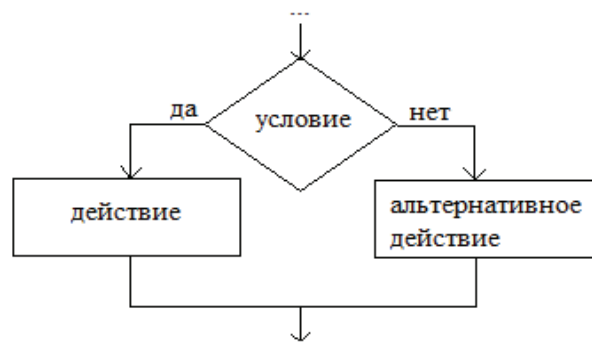


Рисунок 1.3.1 – Графическая схема условного оператора `if`

Условный оператор `if` используется для выбора одного из двух направлений хода программы (последовательности команд, которые должны быть выполнены). *Условный оператор* – это средство ветвления вычислительного процесса.

Условный оператор имеет полную и краткую формы.

Полная форма:

```
if условие then
  begin
    команды1, если условие верно
  end
else
  begin
    команды2, если условие ложно
  end;
```

Неполная (краткая) форма:

```
if условие then
  begin
    команды1, если условие верно
  end;
```

В переводе `if` – если, `then` – тогда, `else` – иначе; `команды1` – команды, которые выполняются если условие истинно; `команды2` – команды, которые выполняются если условие ложно.

Внимание! Если в группе `команды1` или в группе `команды2` более одной команды, то данные команды берутся в программные скобки (операторные):

```
begin
  команды
end;
```

Условие – выражение логического типа, которое может принимать одно из двух значений: `true` (истина – условие выполняется) или `false` (ложь – условие не выполняется). Условия бывают простыми и составными. В *простых условиях* используется только один из знаков: `<`, `>`, `<=`, `>=`, `<>`, `=`.

Пример: `x>0`, `x+y <>6` и т.д.

Составные условия – это несколько простых условий в скобках, соединенных словами `and` («и» обозначает, что должно выполняться и первое действие и второе), `or` («или» обозначает, что должно выполняться хотя бы одно условие).

Примеры:

```
(a>b) and (a>c); (x*x+y*y=r*r) or (x=0);
(a>b) and (a>c) and (a>d) and (a<n) or (a=0).
```

Внимание. Заметьте, что каждое простое условие в составе составного берется в скобки. Простых условий в составе составного может быть сколько угодно.

Операторы `if` могут быть *вложенными*, т.е. в качестве действия и/или альтернативного действия может выступать еще один оператор `if`:

```
if условие1 then if условие2 then действие2_1
                  else действие2_2
                  else if условие3 then действие3_1
                       else действие3_2
```

Оператор выбора.

Иногда требуется осуществить выбор более чем из двух условий. В этом случае применяется *оператор множественного выбора*, позволяющий выбрать из списка одно из условий. Оператор выбора выполняет одно действие из нескольких в зависимости от значения некоторого выражения, называемого *переключателем*. Он имеет следующий вид:

```
case переключатель of
  список выбора1: оператор1;
...
  список выбораN: операторN;
else оператор0
end;
```

Переключатель представляет собой выражение порядкового типа. Как и в операторе `if`, ветвь `else` может отсутствовать.

Оператор `case` (означает «в случае») работает следующим образом: в списке выбора проверяется на совпадение текущее значение переключателя и меток списка, и если найдено совпадение, то выполняется оператор, соответствующий данной строке списка. Если же значение переключателя не найдено ни в одной строке, то выполняется оператор `else`. Если ветвь `else` отсутствует, оператор `case` не выполняет никаких действий, а управление передается внешнему оператору, следующему за конструкцией `case`.

Пример:

```
var m:integer;
begin
  write('Введите номер масти'); readln(m);
  case m of
    1 : writeln('пики');
    2 : writeln('трефы');
    3 : writeln('бубны');
    4 : writeln('червы');
    else writeln('Введите правильный номер');
  end;
end.
```

Список выбора состоит либо из одной константы, либо из диапазона значений вида `a..b` (константа `a` должна быть меньше константы `b`). Можно также перечислить несколько констант или диапазонов через запятую:

```
case DayOfWeek of
  1..5: writeln ('Рабочий день');
```



```
6..7: writeln ( 'Выходной день' );
end;
```

В списках выбора диапазоны меток не должны пересекаться. Например, в результате выполнения следующего фрагмента:

```
case i of
1, 7: writeln(1);
5..9: writeln(2);
end;
```

Программа завершится сообщением об ошибке: пересечение диапазонов меток в операторе case.

Логические операции.

К логическим относятся бинарные операции and, or и xor, а также унарная операция not. Эти операции выполняются с использованием операндов типа boolean и возвращают значения типа boolean. Выражение, имеющее тип boolean, называется *логическим (булевым)*. Выражения в программах могут конструироваться с помощью булевых операций. Эти операции используют понятия алгебры логики, разработанной британским математиком Джорджем Булем.

Операция and-конъюнкция (логическое умножение, «и»). Выражение A and B дает значение true только в том случае, если A и B имеют значение true, в остальных случаях – false: истина тогда и только тогда, когда истины оба операнда.

Операция or-дизъюнкция (логическое сложение, «или»). Выражение A or B дает значение false в том и только в том случае, если A и B имеют значения false, в остальных случаях – результат true: ложно тогда и только тогда, когда ложны оба операнда.

Операция not-инверсия (логическое отрицание, «не»). Выражение not a имеет значение, противоположное значению a: меняет значение логической величины на противоположное.

Операция исключаящее «или». Выражение A xor B дает значение true тогда, когда только одно из A или B имеют значение true, в остальных случаях – результат false.

Результаты операций над логическими данными приведены в таблице 3.1.1.

Таблица 3.1.1

A	B	Not A	A and B	A or B	A xor B
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

Контрольные вопросы и задания:

1. Для чего предназначен условный оператор?
2. Каково назначение оператора выбора? В каких случаях он используется?
3. Каково назначение оператора перехода?

4. Что такое метка и для чего она предназначена?
5. Приведите примеры логических выражений с использованием арифметических операций.
6. Приведите примеры условного оператора в различных формах.
7. Особенности логических операций.

Тема 4. Операторы организации циклов

Оператор в программе – это единое и неделимое предложение, выполняющее какое-либо действие. Типичный простой оператор – это оператор присваивания. Другим примером может служить вызов какой-либо процедуры в программе. Возможно, что под любым оператором подразумевается действие (присваивание, вызов подпрограммы и т. п.). Два последовательных оператора обязательно должны разделяться точкой с запятой «;».

Пример простых операторов:

```
a:=10;  
b:=a*5;  
write (a, b);
```

Если какое-то действие мыслится как единое, но реализуется несколькими различными операторами, то последние могут быть представлены как *составной оператор*. Он предназначен для объединения нескольких операторов в один. *Составной оператор* – это последовательность операторов, перед которой стоит слово `begin`, а после `end`. Слова `begin` и `end`, как отмечалось ранее, часто именуют *операторными скобками*.

Пример составного оператора:

```
begin  
  a:=10;  
  b:=a*5;  
  write (a, b);  
end;
```

Составной оператор может содержать любое количество простых операторов. Он допускает вложенность, т. е. может содержать внутри себя другие составные операторы.

Замечание: составной оператор изменяется в тех случаях, когда синтаксиса языка Pascal допускает использование только одного оператора, в то время как алгоритм требует задания некоторой последовательности действий. В Pascal все управляющие структуры (оператора) различают простой и составной оператор: там, где стоит простой оператор, можно поставить и составной.

Цикл с известным количеством повторений FOR.

Цикл – это такая форма организации действий, при которой многократно повторяется одно действие или серия действий в зависимости от условия. Каждое повторение цикла называется *итерация*.

Цикл с известным количеством повторений называется циклом с параметром. Цикл *for* применяется в том случае, если некоторую последовательность действий надо выполнять несколько раз, причем число повторений заранее известно.

Графическая схема цикла *for* представлена на рисунке 1.4.1.

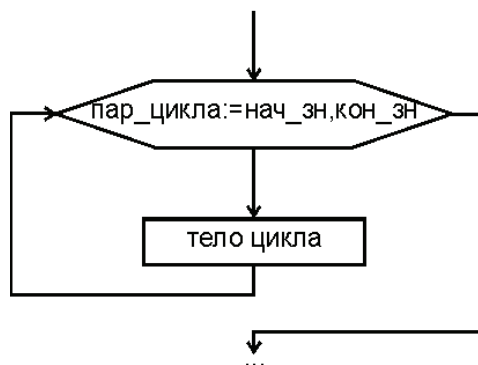


Рисунок 1.4.1 – Графическая схема цикла *for*

Синтаксис цикла с параметром:

```
for переменная:=нач_значение to конеч_значение do
оператор;                               //шаг +1
```

Если в цикле требуется выполнять более одного оператора:

```
for переменная:=нач_значение to конеч_значение do
begin
оператор1;
оператор2;
...
операторN;
end;
```

или

```
for переменная:=нач_значение downto конеч_значение do
оператор;                               //шаг -1
```

Например:

for i:=n1 to n2 do	for i:= n2 downto n1 do
begin	begin
операторы	операторы
end;	end;

Команды, расположенные между операторными скобками *begin..end*, повторяются $n1-n2$ раз, а переменная цикла *i* принимает все значения от $n1$ до $n2$ с шагом 1 ($n1, n1+1, n1+2, \dots, n2$) или от $n2$ до $n1$ с шагом -1.

Переменная *i* называется *счетчиком цикла*, ее значение автоматически изменяется на 1 или -1.

Замечание: все переменные, указанные в строке *for* целого типа (integer). Шаг переменной цикла – только целый (1 или -1). Нельзя изменять значение переменной цикла внутри цикла (например, $i:=i+1$).

Пустой оператор.

Рассмотрим пример:

```
s:=0;
p:=1;
for := 1 to 10 do
  begin          // открывающая операторная скобка
  p:= p*i;
  s:=s+p
  end;          // закрывающая операторная скобка
```

Так как служебное слово `end` является закрывающей операторной скобкой, оно одновременно указывает и конец предыдущего оператора. Поэтому ставить перед ним символ «;» необязательно. Перед `end` может ставиться «;»: в этом случае считается, что последним оператором перед `end` является *пустой оператор*, не выполняющий никаких действий.

Пустой оператор не содержит никаких действий, просто в программу добавляется лишняя «;». В основном пустой оператор и используется для передачи управления в конец составного оператора.

Примеры:

1. *Рассчитать сумму первых натуральных чисел от 1 до 100.*

Подумаем, как бы мы считали устно: $1+2+3+4+5+6+\dots+100$. Самый прямой способ: $1+2=3$, $3+3=6$, $6+4=10$, $10+5=15$, $15+6=21$ и т.д. Т.е. каждый раз к предыдущему значению суммы прибавляем следующее слагаемое.

```
Program primer 1;
Var i, s: integer;
Begin
s:=0;
for i:=1 to 100 do
s:=s+i;
writeln ('s=' , s);
end.
```

Сначала $S=0$. Переменная i принимает значения от 1 до 100 (перебирается каждое слагаемое), которое прибавляется к предыдущему значению суммы: $s:=s+i$. Так как в цикле одна команда, то операторные скобки можно опустить. Поэтому оператор вывода ответа на экран `writeln('S=', s)` находится вне цикла и выполнится один раз.

Заметим, что всегда при выполнении суммы исходное значение суммы принимается равным 0. При вычислении произведения равно 1 (т. к. 0 не изменяет значение суммы, а 1 – значение произведения).

2. *Рассчитать сумму четных натуральных чисел от 7 до n.*

Четные числа от 7 (8, 10, 12, 14) изменяются с шагом 2. Но переменная цикла `for` может изменяться только с шагом 1. Что делать? Добавляем условие.

```

program primer2;
var i,n,s:integer;
begin
write('Введите N=');
readln(n);
s:=0;
for i:=7 to n do
if i/2=int(i/2) then s:=s+i;
writeln('S=',s);
end.

```

Условием `if` из всех чисел от 7 до `n` выбираем только те, которые делятся на 2 – четные. При этом внутри цикла остается одна команда `if`, поэтому операторные скобки можно не ставить.

Цикл с неизвестным количеством повторений.

В цикле с неизвестным количеством повторений вместо счетчика цикла используется условие выхода из цикла.

Цикл `while` является *циклом с предусловием*, т. е. сначала проверяется условие цикла и только в том случае, если оно истинно, выполняется тело цикла.

Графическая схема цикла `while` представлена на рисунке 1.4.2.

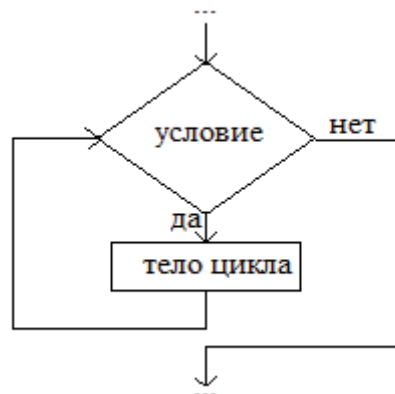


Рисунок 1.4.2 – Графическая схема цикла `while`

Оператор цикла **while** имеет следующую форму:

```
while условие do оператор
```

Эту строку можно интерпретировать как: пока выполняется <условие> выполнять <оператор>.

```

while условие do
begin
оператор1;
...
операторN;
end;

```

Условие представляет собой выражение логического типа и может быть простым (с использованием операций отношения (<, >, <=, >=, =, <>)) или составным (с использованием логических операций and, or).

Оператор после служебного слова do называется *телом цикла*.

Перед каждой итерацией цикла условие проверяется. Если оно оказывается истинно, то выполняется тело цикла. В противном случае происходит выход из цикла.

Если условие всегда оказывается истинным, то может произойти заикливание:

```
while 2>1 do
write (1);
```

Для выхода из заикливания программы можно использовать комбинацию клавиш Ctrl-F2 или кнопку Stop.

Цикл *repeat* является *циклом с постусловием*, т. е. сначала выполняется тело цикла, затем проверяется условие цикла. Пока оно не выполнится, (т.е. пока условие ложно) цикл будет повторяться.

Графическая схема цикла *repeat* представлена на рисунке 1.4.3.

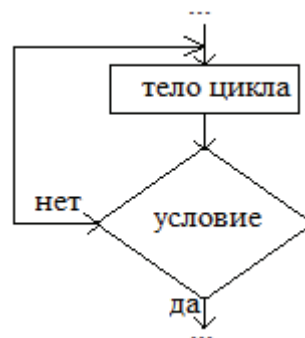


Рисунок 1.4.3 – Графическая схема цикла *repeat*

Оператор цикла **repeat** имеет следующую форму:

```
repeat оператор until условие
```

Эту строку можно понимать как: повторять <операторы> до тех пор, пока не выполнится <условие>.

```
repeat
оператор1;
...
операторN
until условие;
```

В отличие от цикла *while*, условие вычисляется после очередной итерации цикла. Если оно истинно, то происходит выход из цикла.

Таким образом, операторы, образующие *тело цикла* оператора *repeat*, выполняются, по крайней мере, один раз.

Если условие всегда оказывается ложным, то может произойти заикливание:

```
repeat
write (1)
until 2=1;
```

Для выхода из заикливания программы можно использоваться комбинацию клавиш Ctrl-F2 или кнопку Stop.

Из-за того, что в циклах этого типа выполнение цикла происходит раньше проверки условия, в них бывает трудно найти ошибки. Поэтому предпочтительнее использовать циклы с предусловием, тем более, что ими всегда можно заменить циклы с постусловием.

Существуют операторы прерывания выполнения цикла [1]:

`continue` – оператор завершения шага цикла. По этому оператору прекращается выполнение текущего шага цикла, и управление передается на следующий шаг;

`break` – оператор «выхода» из цикла. Прекращается выполнение цикла и управление передается оператору, следующему за последним оператором цикла;

`exit` – оператор «выхода» из программы. По этому оператору выполнение программы завершается. Рекомендуется использовать для выхода из подпрограмм.

Пример:

Составить программу, которая находит наибольший общий делитель (НОД) двух данных целых чисел и выводит на экран результат.

Найдем НОД двух чисел с помощью следующего алгоритма: будем из большего числа вычитать меньшее до тех пор, пока они не станут равными. Полученное число и есть результат. Такой алгоритм называется *алгоритмом Евклида*.

```
program nod_while;
var a, b: integer;
begin
write ('a= ');
readln(a);
write ('b= ');
readln(b);
while a<>b do           {проверка равенства чисел}
if a>b then
a:=a-b
else
b:=b-a;                {всегда из больше вычитаем мень-
ше}
writeln('НОД= ',a)     {после выхода из цикла a=b – ис-
комый результат}
end.
```

Рассмотрим уже известный нам алгоритм Евклида с оператором `repeat`.

```

program nod_repeat;
var a, b: integer;
begin
write ('a= ');
readln(a);
write ('b= ');
readln(b);
repeat
if a>b then
a:=a-b
else
b:=b-a
until a=b;
writeln('НОД= ', a)
end.

```

Контрольные вопросы и задания:

1. Назовите особенности операторов организации циклов.
2. Какие типы данных могут быть использованы для счетчика цикла for?
3. В чем отличие между операторами for..to и for..downto?
4. Чем отличается цикл while от цикла repeat?
5. Когда тело цикла while ни разу не выполняется?
6. Назовите операторы прерывания выполнения цикла.

Тема 5. Массивы: определение, описание, размещение в памяти, использование

Массив – это упорядоченная последовательность данных одного типа, рассматриваемых как единое целое.

Каждое из значений, составляющих массив, называется его *компонентой* (или *элементом* массива). Элементы массива могут быть целого, вещественного, строкового, символьного и других типов.

Доступ к элементам массива осуществляется по *индексу* (порядковому номеру).

Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом.

Размерность массива – количество индексов элементов массива. По размерности массивы делятся на одномерные (линейные), двумерные, трехмерные и т.д.

Размер массива – количество элементов в нем.

Количество элементов массива задается при его *описании*.

При *описании массива* ему присваивается имя, пишется служебное слово `array`, указывается число входящих в массив элементов [...] и тип этих элементов. В языке Pascal массив может быть описан одним из следующих способов:

I способ:

```
type имя типа=array [тип индекса] of тип элементов;  
var идентификатор: имя типа;
```

II способ:

```
var идентификатор: array [тип индекса] of тип элементов;
```

Если массивы одного типа и одинакового диапазона, их можно объявить списком.

Например:

```
var  
a,b: array [1..15] of integer;  
mass1: array [20..50] of real;  
massiv: array [0..255] of real;
```

Диапазон массива задается левой и правой границами изменения индекса массива, так что, массив `a` состоит из 16-и элементов, `mass1` – из 31, а массив `massiv` – из 256 элементов.

Пример объявления одномерного массива в разделе переменных:

```
const n=100;  
var A: array[1..n] of real;
```

Пример объявления одномерного массива в разделе типов:

```
const n=100;  
type mas=array[1..n] of integer;  
var A: mas;
```

Элементы массива называются *индексированными переменными*, они могут быть использованы в выражениях как простые переменные. Элементы массива располагаются в памяти последовательно.

В случае, когда левая граница диапазона равна 1, индекс элемента совпадает с его порядковым номером.

В программе имя любого элемента массива состоит из *имени массива* и *индекса элемента* в квадратных скобках.

Например: `a[5]`, `mass1[3]` и `massiv[256]`.

Массивы бывают одномерные и многомерные.

Массив бывает одномерным (линейным), если в каждом из его элементов имеется только один индекс.

К примеру, если будем в течение месяца ежедневно записывать среднесуточную температуру и заносить эти данные в таблицу, то получится одномерный массив, в котором будет храниться переменная `t[j]` с одним индексом (номером столбца).

Город	t[1]	t[2]	t[3]	t[...]	t[30]
Могилев	15	17	14	...	21

$t[1], t[2], t[3], \dots, t[n]$.

В двумерном массиве у каждого элемента имеется два индекса.

Пример объявления двумерного массива в разделе переменных:

```
const m=30; n=50  
var A: array[1..m, 1..n] of char;
```

Пример объявления двумерного массива в разделе типов:

```
const m=30; n=30;  
type matr=array[1..m, 1..n] of longint;  
var A: matr;
```

Двумерный массив с равным количеством строк и столбцов называется *квадратной матрицей*.

Если занести в таблицу данные о температуре в нескольких городах, то получится *двумерный массив*, где в каждой ячейке будет храниться переменная с двумя индексами $t[i, j]$, где i – номер строки, а j – номер столбца.

Могилев	15	17	14	...	21
Минск	16	15	17	...	20
Брест	14	12	16	...	18

Матрицы могут быть *прямоугольными* или *квадратными*. Именно квадратные матрицы наиболее интересны для решения задач на программирование, в связи с тем, что у них имеются главная и вспомогательная диагонали. На главной диагонали лежат элементы, у которых $i=j$. На вспомогательной диагонали – элементы, у которых $i=n-j+1$, где n – количество строк (столбцов).

Ввод массивов можно производить вручную (с клавиатуры), или автоматически, с помощью генерирования случайных чисел `random`, из файла или вычислить по формуле. Обработку массива производят с помощью циклов.

Способы заполнения (ввода) массива данными:

С клавиатуры:

```
for i:=1 to 40 do read(a[i]);
```

или для двумерного массива

```
for i:=1 to 10 do  
for j:=1 to 10 do  
read(a[i, j] );
```

По формуле:

```
for i:=1 to 40 do  
a[i]:=i*i+5;
```

или для двумерного массива

```
for i:=1 to 10 do  
for j:=1 to 10 do  
a[i, j]:=i*j-4;
```

С помощью генератора случайного числа:

```
for i:=1 to 40 do  
a[i]:=random(20);
```

или для двумерного массива

```
for i:=1 to 10 do
for j:=1 to 10 do
a[i,j]:=random(20);
```

В языке программирования Pascal для генерации случайных чисел в заданных диапазонах используется функция `random`. Перед ее использованием обычно выполняется процедура инициализации датчика случайных чисел – `randomize`; иначе программа всегда будет выдавать один и тот же результат. `Randomize` задает начальное значение последовательности, от которого вычисляются все последующие. При каждом запуске программы это значение будет разным, а значит и результат работы функции `random` будет различным.

Функция `random` генерирует случайное число в диапазоне от 0 (включительно) до 1. Если в скобках указан аргумент, то 0 до значения, указанного в скобках (не включая само значение). Так, выражение `random(20)`, говорит о том, что будет получено число в диапазоне $[0,20)$. Если требуется получать значения в каком-либо другом диапазоне (не от нуля), то прибегают к математической хитрости. Например, чтобы получить случайное число от -100 до 100 достаточно записать такое выражение: `random(200)-100`. В результате, сначала будет получено число из диапазона $[0,199]$, а затем из него будет вычтена сотня.

Пример:

```
var n,i,x: integer;
begin
randomize;
n:=random(7)+5;
for i:=1 to n do
begin
x:=random(100)-50;
write(x:5);
end;
end.
```

В примере программы сначала с помощью процедуры `randomize` инициализируется датчик случайных чисел. Далее переменной `n` присваивается случайное значение в диапазоне $[5,12)$. Значение переменной `n` используется для определения количества итераций цикла `for`. В цикле `for` генерируются случайные числа в диапазоне $[0,50)$ и выводятся на экран.

Вывод элементов массива на экран монитора:

```
for i:=1 to 40 do write(a[i]);
или для двумерного массива (в виде матрицы)
for i:=1 to 10 do
begin
for j:=1 to 10 do
write(a[i,j] );
writeln;
end;
```

Одна из типовых задач при обработке одномерных массивов – *поиск минимально и максимального элементов массива* (например, минимальная и максимальная температура).

Пример:

Одномерный массив заполнен целыми числами. Отсортируйте его по убыванию, выведите на экран max и min элементы.

```
program sort;
var a: array [1..50] of integer;
t,f,n,i,max,k: integer;
begin
write ('Введите количество элементов массива:'); //
ввод массива
readln (n);
for i:=1 to n do
begin
write ('A[' ,i, '=');
readln (a[i]);
end;
for t:=1 to n-1 do //сортировка: поиск max и его
порядкового номера
begin
max:=a[t];
k:=t;
for i:=t+1 to n do
if a[i]<max then
begin
max:=a[i];
k:=i;
end;
f:=a[t]; //перестановка элементов массива с max
a[t]:= a[k];
a[k]:=f;
end;
writeln ('Максимальный элемент - ',a[i]); // Вывод
элементов
writeln ('Минимальный элемент - ',a[1]);
end.
```

Контрольные вопросы и задания:

1. Что такое массив?
2. Назовите способы описания массивов.
3. Могут ли в массиве храниться целые и вещественные числа вперемешку?
4. Может ли индекс массива быть вещественным числом? Может ли быть тип индекса целочисленным?
5. Какой оператор цикла наиболее естественен для обработки массивов?
6. Назовите способы заполнения.
7. В чем отличия одномерного от двумерного массива?
8. Особенности использования функции random?

Тема 6. Процедуры и функции

Во многих программах встречаются группы одинаковых команд, которые выполняются много раз. Чтобы упростить вид программы, наборам таких команд присваиваются имена и, когда требуется выполнить эти группы команд, указывают *только их имена* (вызывают по имени). Такие группы команд называют *подпрограммами*. Применение подпрограмм дает возможность уменьшать число повторений одной и той же последовательности операторов, а также конструировать программу как набор отдельных подпрограмм.

Подпрограмма – это программа, реализующая вспомогательный алгоритм. *Основная программа* – это программа, реализующая основной алгоритм решения задачи и содержащая в себе обращение к подпрограммам. В Pascal существует два типа подпрограмм:

- подпрограмма-процедура;
- подпрограмма-функция.

Функциями называют такие группы команд, которые при своем выполнении производят какие-либо вычисления и соответственно возвращают какое-то значение. *Процедурами* называют какие-либо действия, которые выполняет программа (выводит данные на экран, удаляет символ строки и т. д.).

Все процедуры и функции делятся на две группы: стандартные и пользовательские (создаваемые разработчиками программы).

Стандартные процедуры и функции входят в состав языка и вызываются для выполнения по своему имени. Процедура или функция представляет собой последовательность операторов, которая имеет имя, список параметров и может быть вызвана из различных частей программы.

И процедура, и функция должны иметь собственное имя и может содержать произвольное число операторов и даже внутренних процедур и функций. Любая используемая в программе процедура или функция должна быть описана *в разделе описаний*.

Операторы подпрограммы, выделенные операторными скобками (begin..end), называются *телом подпрограммы*.

Структура описания процедуры и функции [1,5]:

Подпрограмма-процедура
procedure <имяПроцедуры>
(<список формальных параметров>);
локальные объявления
begin
тело процедуры
end.

Подпрограмма-функция
function <имяФункции>
(<список формальных параметров >):<тип результата>;
локальные объявления
begin
тело функции
имя Функции:=выражение;
end.

Обращение к функции является операндом в выражении. В качестве результата процедура может возвращать в вызывающую программу множество простых или структурированных величин или не возвращать никаких значений.

Среди формальных параметров выделяют *параметры-значения* и *параметры-переменные*. Параметры-значения используются для передачи в процедуру исходных данных. Они указываются через запятую. Параметры-переменные необходимы для сохранения в них результата выполнения процедур. Они записываются после служебного слова `var`.

Переменные, описанные в разделе описаний основной программы и используемые в основной программе и в подпрограммах, называются *глобальными*, а переменные, используемые только в подпрограмме и описанные в разделе описаний переменных подпрограммы, называются *локальными*. Обмен данными между вызывающей программой и подпрограммой может происходить не только через параметры, но и через глобальные переменные.

Вызов процедуры из основной программы выполняется следующим образом: имя процедуры (список фактических параметров).

Между формальными и фактическими параметрами должно существовать соответствие по количеству параметров, порядку их следования и типу данных. Фактические параметры-переменные и соответствующие им формальные параметры называются *выходными параметрами процедуры*. А фактические параметры-значения и соответствующие им формальные параметры называются *входными параметрами процедуры*.

После завершения выполнения процедуры управление передается в основную программу оператору, который первым следует за вызовом процедуры.

Функция имеет список формальных параметров-значений, которые являются входными параметрами. Имя функции является одновременно и результатом ее выполнения, поэтому в заголовке необходимо указывать тип результата.

Отличительными особенностями функции от процедуры является наличие у функций только одного результата выполнения, а также то, что вызов функции осуществляется внутри выражения по ее имени.

Пример:

Вычислить разность двух простых дробей $a/b-c/d=e/f$, где a,b,c,d,e,f – натуральные числа. Результат получить в виде несократимой дроби e/f .

Следует вычислить числитель и знаменатель по правилам вычитания дробей и сократить их на наибольший общий делитель (НОД). Вычисление НОД двух чисел оформим в виде подпрограмм, используя алгоритм Евклида. Рассмотрим два варианта решения этой задачи.

С помощью подпрограммы-функции:

```
program s1;
var a,b,c,d,e,f: integer;
function NOD(n,m:integer): integer;
begin
while m<>n do
if m>n then m:= m-n else n:=n-m;
```

```

NOD:= m;
end;
begin
write ('Введите числители и знаменатели дробей:');
readln (a,b,c,d);
e:= a*d-b*c;
f:=b*d;
if e=0 then writeln (c)
else
begin
e:= e div NOD (abs(e), f);
f:= f div NOD (abs(e), f);
writeln (a,'/',f);
end;
end.

```

С помощью подпрограммы-процедуры:

```

program s;
var a,b,c,d,e,f,g: integer;
procedure NOD(n,m:integer;var k:integer);
begin
while m<>n do
if m>n then m:= m-n else n:=n-m;
k:= m;
end;
begin
write ('Введите числители и знаменатели дробей:');
readln (a,b,c,d);
e:= a*d-b*c;
f:=b*d;
if e=0 then writeln (c)
else
begin
NOD (abs(e), f,g);
NOD(abs(e), f,g);
e:= e div g;
f:= f div g ;
writeln (a,'/',f);
end;
end.

```

Контрольные вопросы и задания:

1. Что такое подпрограмма?
2. Зачем из базового алгоритма программы выделяют подпрограммы?

3. Чем подпрограмма отличается от основной программы?
4. Особенности подпрограммы-функции и подпрограммы-процедуры?
5. В чем отличие формальных параметров от фактических?
6. Приведите примеры стандартных процедур и функций.

Тема 7. Символьные переменные и строки

Символьная (литерная) константа – это любой символ языка, заключенный в апострофы.

Примеры: 'd', '5', '+', '']'.

Символьная константа занимает один байт памяти. Символьная переменная типа `char` – это переменная, принимающая значение символьной константы.

Символьные переменные описываются следующим образом:

```
var идентификатор: char;
```

Например, `var a, d: char;`

Все символы языка Pascal упорядочены, т.е. каждый символ имеет свой порядковый номер (таблица кодов символов ASCII). Это позволяет применять к символьным данным операции сравнения: `<`, `>`, `=`, `<>`, `>=`, `<=`. Например, результат операции сравнения 'A' < 'B' будет истинным, так как сравниваются их порядковые номера, а они равны 66 и 67 соответственно.

В Pascal допускается использование последовательности символов, заключенной в апострофы, длиной не более 256 символов. Например: 'grupa', 'Группа 2ПО' и т.д. Такие последовательности относятся к строковым типам данных. Pascal предоставляет средства для работы с данными строкового типа. Строковый тип данных представляет собой цепочку символов. Длина цепочки может изменяться от 0 до 255.

Для определения строкового типа используется служебное слово `string`, за которым в квадратных скобках указывается максимальная длина строки.

Строки в языке Pascal можно описать одним из следующих способов:

I способ:

```
type имя типа= string [n];  
var идентификатор: имя типа;
```

II способ:

```
var идентификатор: string [n];
```

Здесь `n` – количество символов в строке. Если максимальный размер строки не указан, то он автоматически принимает значение 255.

Например:

```
type krug= string[30];  
var mkrug: krug;
```

...

В данном примере переменная `mkrug` представляет собой последовательность, включающую до 30 символов, причем каждый символ имеет стандартный тип `char`.

Значение строковой переменной может быть назначено оператором присваивания или введено с устройства ввода.

Например:

```
arug:='диам';  
krug:=arug;  
readln(krug);
```

Изображение строки строится из цепочки символов и заключается в апострофы.

Для строковой переменной длиной n символов отводится $n+1$ байтов памяти, из которых n байтов предназначены для хранения символов строки, а один байт – для хранения текущей длины строки.

Символы строки нумеруются целыми числами, начиная с единицы, а байт с текущей длиной строки считается нулевым ее элементом.

Если присваивается значение строкового выражения с длиной большей, чем максимально допустимая, происходит отсечение строки до максимальной длины.

Например:

```
var st: string[7];  
begin  
st:='длинная строка';  
writeln(st); //будет отображено только: 'длинная'}  
...
```

Для строковых типов данных определена операция «конкатенация», обозначаемая символом `+`. Смысл операции заключается в формировании новой строки. Динамическая длина сформированной строки равна сумме символов строк-операндов, а ее значение равно последовательности символов исходных строк.

Например:

```
var str1, str2 : string[10];  
st: string[25];  
begin  
str1:='Паскаль -';  
str2:='программа';  
st:=str1+str2;  
writeln(st)  
end.
```

В результате выполнения программы будет на экране отображена текстовая строка: 'Паскаль – программа'.

Кроме операции конкатенации над значениями строковых типов разрешены операции сравнения <, <=, >, >=, =, <>. Более короткая строка всегда меньше более длинной. Если длины сравниваемых строк равны, то происходит поэлементное сравнение символов этих строк: сравнение строк выполняется до первого не совпадающего символа, и большей считается та строка, у которой не совпадающий символ имеет больший код.

Доступ к отдельным элементам строк осуществляется аналогично доступу к элементам одномерного массива: после имени строковой переменной необходимо в квадратных скобках указать арифметическое выражение целого типа.

Можно заметить, что работа со строковыми данными аналогична работе с символьными массивами, однако, это не означает их полную идентичность. Так, распространенной ошибкой является работа с элементами строки без учета ее текущей длины.

Стандартные функции для данных типа **char**.

Таблица 7.1.1.

Назначение функции	Заголовок функции	Пример	Результат
Преобразование выражения типа байт в символ.	<code>chr(x:byte):char;</code>	<code>x:=65; a:=chr(x); writeln(a);</code>	'А'
Преобразование изображения символа в код типа байт.	<code>ord(a:char):byte;</code>	<code>a:='А'; x:=ord(a); writeln(x);</code>	65
Возвращение предыдущего символа.	<code>pred(x:char):char;</code>	<code>a:='Б'; b:=pred(a); writeln(b);</code>	'А'
Возвращение следующего символа.	<code>succ(x:char):char;</code>	<code>a:='К'; b:=pred(a); writeln(b);</code>	'Й'
Преобразование строчной буквы в прописную букву.	<code>Uppcase(x:char):char;</code>	<code>a:='d'; b:=Uppcase(a); writeln(b);</code>	'D'

Стандартные процедуры и функции для данных типа **string**.

Таблица 7.1.2

Назначение функции	Заголовок функции	Пример	Результат
Сцепление строк St1, St2, ..., Stn.	<code>concat (St1,St2,...,Stn:string):string;</code>	<code>St1:='За'; St2:='дача'; y:=concat(St1,St2); writeln(y);</code>	'Задача'
Выделение из строки st подстроки длиной n символов, начиная с позиции p.	<code>copy (st:string,p,n:byte):string;</code>	<code>st:='бегемот'; d:=copy(st,5,3); writeln(d);</code>	'мот'

Назначение функции	Заголовок функции	Пример	Результат
Вычисление длины строки st.	<code>length (st:string):byte;</code>	<code>st:='школа'; d:=length(st); writeln(d);</code>	5
Определение позиции первого появления в строке st2 подстроки st1.	<code>pos (st1,st2): byte;</code>	<code>st2:='колледж' ; st1:='л'; d:=pos(st1,st2); writeln(d);</code>	3

Таблица 7.1.3

Назначение процедуры	Заголовок процедуры	Пример	Результат
Удаление n символов строки st, начиная с позиции pos.	<code>delete (st:string, pos,n:byte);</code>	<code>st:='антрекот'; delete(st,1,5); writeln(st);</code>	'кот'
Вставка строки str1 в строку str2, начиная с позиции p.	<code>insert (str1,str2: string,p:byte);</code>	<code>st2:='колледж'; st1:='л'; insert(st1,st2,3); writeln(st2);</code>	'колледж'
Преобразование числового значения величины n (integer или real), помещение результата в строку st.	<code>str (n:real, st:string);</code>	<code>x:=1234; y:=5678; str(x,st1); str(y,st2); st:=st1+st2; writeln(st);</code>	'12345678'
Преобразование строки st в величину целочисленного или вещественного типа, помещение результата в n; code – целочисленная переменная, которая равна номеру позиции в строке st, начиная с которой произошла ошибка преобразования. Если преобразование прошло без ошибок, то переменная code равна 0.	<code>val (st:string,n:real, code:integer)</code>	<code>st1:='23.12.1980' ; st2:=copy(st1,7, 4); val(st2,god,code); let:=2018-god; writeln(let,',_c ode=',code);</code>	38,_code=0

Контрольные вопросы и задания:

1. Что такое символьная константа, символьная переменная?
2. Как описываются строки?
3. Какова максимальная длина строки?
4. Каким образом производится сравнение строк?
5. Каким идентификатором определяются данные строкового типа?
6. Как происходит сравнение символов и строк?

7. Каково назначение специальных процедур и функций обработки данных символьного и строкового типов?

Тема 8. Организация библиотек. Стандартные библиотечные модули и модули пользователя. Структура Unit-а

Иногда в разных местах программы приходится выполнять практически одни и те же последовательности действий с разными исходными данными. Такие последовательности действий можно оформить в виде так называемых *подпрограмм* (от англ. *subroutine*) – сгруппировать операторы в блок, к которому можно обратиться по имени, причем неоднократно (тема 6). Подпрограммы сокращают текст программы, существенно уменьшают время их исполнения, облегчают жизнь программистам, которые могут создавать программы *модульно*.

Набор подпрограмм, которые могут быть использованы при разработке целого ряда программ, удобно оформить в виде отдельных тематических *библиотек*. Для организации таких библиотек в Pascal введены *модули*.

Модули в Pascal по отношению к основной части программы напоминают подпрограммы (процедуры и функции). Но по определению они являются *самостоятельными* программами, ресурсы которых могут быть задействованы в других программах. Кроме того, описание модулей происходит в отдельном файле, поэтому *модуль* – это отдельно компилируемая программа.

История концепции модулей.

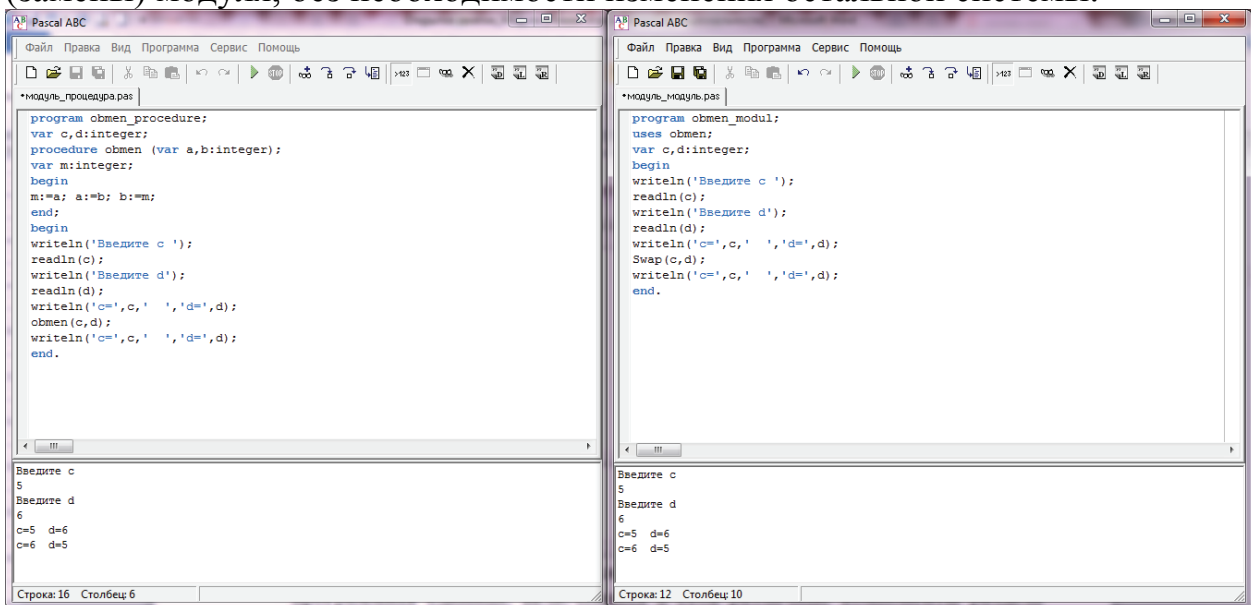
Термин «модуль» в программировании начал использоваться в связи с внедрением модульных принципов при создании программ. В 70-х гг. под модулем понимали какую-либо процедуру или функцию, написанную в соответствии с определенными правилами. Первым основные свойства программного модуля более-менее четко сформулировал Дэвид Парнас. Парнас первым выдвинул концепцию скрытия информации в программировании. Однако существовавшие в языках 70-х гг. только такие синтаксические конструкции, как процедура и функция не могли обеспечить надежного скрытия информации, поскольку подвержены влиянию глобальных переменных, поведение которых в сложных программах бывает трудно предсказуемым. Решить эту проблему можно было только разработав новую синтаксическую конструкцию, которая не подвержена влиянию глобальных переменных. Такая конструкция была создана и названа *модулем*. Изначально предполагалось, что при реализации сложных программных комплексов модуль должен использоваться наравне с процедурами и функциями как конструкция, объединяющая и надежно скрывающая детали реализации определенной подзадачи.

Впервые специализированная синтаксическая конструкция модуля была предложена Н. Виртом в 1977 г. и включена в его новый язык Modula. По своей организации и характеру использования в программе в модулях Паскаля явным образом выделяется некоторая «видимая» интерфейсная часть, в которой скон-

центрированы описания глобальных типов, констант, переменных, а также приводятся заголовки процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполняемой части модуля, которая может быть скрыта от пользователя.

Модули представляют собой прекрасный *инструмент для разработки библиотек прикладных программ и мощное средство модульного программирования*. Модули создаются, как правило, для обеспечения компактности кода, о чем приходится заботиться крупным проектам.

Принцип модульности является средством упрощения задачи проектирования ПС и распределения процесса разработки ПС между группами разработчиков. При разбиении ПС на модули для каждого модуля указывается реализуемая им функциональность, а также связи с другими модулями. Удобство использования модульной архитектуры заключается в возможности обновления (замены) модуля, без необходимости изменения остальной системы.



Модуль – это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые исполняемые операторы иницилирующей части.

Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам.

Структура модуля.

Модуль Pascal имеет следующую структуру:

```
unit <имя_модуля>;  
interface <интерфейсная часть>;  
implementation <исполняемая часть >;  
begin  
<иницилирующая часть>;  
end.
```

`unit` – зарезервированное слово (единица); начинает заголовок модуля;
<имя_модуля> – имя модуля;
`interface` – зарезервированное слово (интерфейс); начинает интерфейсную часть модуля;
`implementation` – зарезервированное слово (выполнение); начинает исполняемую часть модуля;
`begin` – зарезервированное слово; начинает иницилирующую часть модуля; причем конструкция `begin` <иницилирующая часть> необязательна;
`end` – зарезервированное слово – признак конца модуля.

Таким образом, модуль Pascal состоит из *заголовка и трех составных частей*, любая из которых может быть пустой.

Заголовок модуля состоит из зарезервированного слова `unit` и следующего за ним имени модуля. Имя модуля Pascal должно совпадать с именем дискового файла, в который помещается исходный текст модуля. Если, например, имеем заголовок модуля

```
unit obmen;
```

то исходный текст этого модуля должен размещаться на диске в файле `obmen.pas`.

Имя модуля служит для его связи с другими модулями и основной программой. Эта связь устанавливается специальным предложением:

```
uses <список модулей>;
```

Здесь `uses` – зарезервированное слово (перевод – использовать); <список модулей> – список модулей, с которыми устанавливается связь; элементы списка – имена модулей *через запятую*.

Например, `uses crt, obmen;`

Предложение `uses` <список модулей> должно стоять сразу после заголовка программы, т.е. должно открывать раздел описаний основной программы.

В модулях могут использоваться другие модули. В модулях предложение `uses` <список модулей> может стоять сразу после слова `interface`.

Пример фрагмента программы:

```
Unit obmen;  
Interface  
uses crt;  
Procedure Swap (var a,b:integer);  
Implementation
```

Интерфейсная часть открывается зарезервированным словом `interface`. В этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и подпрограмм), которые должны быть доступны основной программе и (или) другим модулям Паскаля. При объявлении глобальных подпрограмм в интерфейсной части указывается только их заголовки, например:

Пример фрагмента программы:

```
Unit obmen;
Interface
uses crt;
Procedure Swap (var a,b:integer);
```

Если теперь в основной программе написать предложение `uses obmen;`, то в программе станет доступна процедура из модуля `obmen`.

Исполняемая часть модуля начинается зарезервированным словом `implementation` и содержит описания подпрограмм, объявленных в интерфейсной части. В ней могут объявляться локальные для модуля объекты – вспомогательные типы, константы, переменные и блоки, а также метки.

Описанию подпрограммы, объявленной в интерфейсной части модуля, в исполняемой части должен предшествовать заголовок, в котором можно опустить список формальных параметров, так как они уже описаны в интерфейсной части. Но если заголовок подпрограммы приводится в полном виде, т.е. со списком параметров и объявлением типа результата для функции, то он должен полностью совпадать с заголовком подпрограммы в интерфейсной части.

Пример модуля:

```
Unit my_arifm;
Interface
function min (a,b:integer):integer;
function max (a,b:integer):integer;
Implementation
function min (a,b:integer):integer;
begin
  if a>b then min:=b
  else min:=a;
end;
function max (a,b:integer):integer;
begin
  if a>b then max:=a
  else max:=b;
end;
end.
```

Иницирующая часть завершает модуль. Она может отсутствовать вместе с начинающим ее словом `begin` или быть пустой – тогда вслед за `begin` сразу следует признак конца модуля.

В иницирующей части размещаются исполняемые операторы, содержащие некоторый фрагмент программы. Эти операторы исполняются до передачи управления основной программе и обычно используются для подготовки ее работы. Например, в иницирующей части могут иницироваться переменные, открываться файлы, устанавливаться связи с другими компьютерами и т. п.:

Пример модуля:

```
Unit fileText;
Interface
Procedure print(s: string);
implementation
var f: text;
const
name= 'output.txt';
procedure print;
begin
writeln(f, s)
end ;
{начало иницилирующей части}
begin
assign(f, name);
rewrite ( f );
{конец иницилирующей части}
end.
```

Не рекомендуется делать иницилирующую часть пустой, лучше ее опустить.

Примеры возможных ошибок: когда модуль не находится в одной папке с основной программой; когда имя файла модуля отлично от имени модуля; когда в основной программе прописано два раза предложение `uses`.

Алгоритм решения задач с использованием модулей:

- 1) разработать модуль, сохранив его отдельным файлом на диске;
- 2) подключить модуль к основной программе с помощью служебного слова `uses`. Файл основной программы должен находиться на том же диске и в той же папке, что и файл модуля;
- 3) имя файла модуля должно совпадать с именем подключаемого модуля, например, `obmen.pas` и `uses obmen;`

Стандартные модули Pascal

В Pascal имеется ряд стандартных модулей, в которых описано большое количество встроенных констант, типов, переменных и подпрограмм. Каждый модуль содержит связанные между собой ресурсы.

Модуль *System*: модуль содержит базовые средства языка, которые поддерживают ввод-вывод, работу со строками, операции с плавающей точкой и динамическое распределение памяти. Этот модуль автоматически используется во всех программах и его не требуется указывать в операторе `uses`. Он содержит все стандартные и встроенные процедуры, функции, константы и переменные Pascal.

Модули *Dos* и *WinDos*: содержат подпрограммы, реализующие возможности операционной системы MS-DOS – например, переименование, поиск и удаление файлов, получение и установку системного времени, выполнение программных прерываний и т. д.

Модуль Crt: модуль предназначен для организации эффективной работы с экраном, клавиатурой и встроенным динамиком. В нем сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом работы экрана. С его помощью можно перемещать курсор в любую точку экрана, менять цвет выводимых символов и фона, создавать окна.

Модуль Graph: модуль обеспечивает работу с экраном в графическом режиме. Система Pascal обладает великолепной графической библиотекой. С помощью процедур и функций этого модуля можно создавать разнообразные графические изображения. В отличие от текстового режима, в графическом – курсор невидим.

Пример.

Разработать модуль, содержащий арифметические операции над целыми числами: сложение, вычитание и умножение. Разработать программу для нахождения суммы, разности и произведения двух целых чисел, используя разработанный модуль.

Решение:

Модуль:

```
unit m_m;
interface
function sum (a,b:integer):integer;
function minus (a,b:integer):integer;
function mult (a,b:integer):integer;
Implementation
function sum (a,b:integer):integer;
begin
sum:=a+b;
end;
function minus (a,b:integer):integer;
begin
minus:=a-b
end;
function mult (a,b:integer):integer;
begin
mult:=a*b;
end;
end.
```

Сохраняем модуль под именем max_min.pas в одну папку с основной программой.

Программа:

```
program modul;
uses max_min;
var
c,d,z_sum,r_minus,t_mult:integer;
begin
```

```
writeln('Введите c');  
readln(c);  
writeln('Введите d');  
readln(d);  
z_sum:=sum(c,d);  
r_minus:=minus(c,d);  
t_mult:=mult(c,d);  
writeln('z_sum=',z_sum);  
writeln('r_minus=',r_minus);  
writeln('t_mult=',t_mult);  
end.
```

Контрольные вопросы и задания:

1. Что такое модуль?
2. Назовите структуру модуля.
3. Назовите стандартные модули Pascal.
4. Могут ли быть подключены в разрабатываемой единице модули, написанные пользователем самостоятельно?
5. Какое зарезервированное слово отвечает за заголовок модуля?
6. Как подключается модуль к разрабатываемой единице?
7. Где хранится разработанный модуль на диске?

Глава II. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Лабораторная работа № 1

Разработка и отладка линейных алгоритмов и программ

Цель работы: сформировать умения разрабатывать и отлаживать линейные алгоритмы и программы.

Методические указания:

- изучить темы «Понятие алгоритма: свойства, способы описания» и «Структура программного модуля. Состав интегрированной программной среды»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое алгоритм?
2. Что такое программа линейной структуры?
1. Каков порядок создания программы в среде PascalABC.Net?
2. Какова структура Pascal-программы?
3. Как сохранить программу под другим именем?
4. Каков алгоритм работы оператора присваивания?
5. Чем отличаются данные целого типа?
6. Что значит «объявить переменную»?
7. Каким образом определяется порядок выполнения действий в арифметических выражениях?
8. Какие операторы используются в линейных программах?
9. Перечислите стандартные типы данных в Pascal.
10. Какие данные называют арифметическими?
11. Что такое функция и назовите основные стандартные функции языка Pascal?
12. Каков синтаксис оператора ввода и вывода данных?
13. Для чего нужен формат вывода данных? Как он задается?

Задания к работе.

1. Загрузите систему программирования PascalABC.Net.
2. В появившемся окне введите следующий текст:

```
program sum;  
var a,b:byte;  
begin  
write('a=');  
readln(a);  
write('b=');
```

```
readln(b);
writeln('a+b=', a+b);
end.
```

3. Сохраните введенный текст в файле SUM.PAS в своем рабочем каталоге. Выполните программу для $a=2$ и $b=3$.
4. С помощью программы решите следующие примеры:
 - 1) $25+10$;
 - 2) $-25+10$;
 - 3) $25000+10000$;
 - 4) $-25000+10000$.

Получены ли верные результаты? Если «нет», то почему? Какие из приведенных примеров можно будет вычислить, если заменить тип переменных a и b на тип `word`, а какие нельзя и почему? Убедитесь в этом. Измените тип переменных a и b таким образом, чтобы все примеры вычислялись правильно. Сохраните текст полученной программы в новом файле в своем рабочем каталоге.

5. Откройте файл с исходным вариантом программы, сохраненным при выполнении задания 2. Измените программу так, чтобы ввод исходных данных осуществлялся одним оператором и сопровождался соответствующим поясняющим сообщением, а в результате выполнения программы на экран выводилось сообщение: «...+...=...». Вместо многоточий предусмотреть вывод конкретных значений по смыслу (см. рис).

```
Введите числа a и b: 5 6
5+6=11
```

6. Сохраните текст полученной программы. Выполните полученную программу для $a=2$ и $b=3$, разделяя вводимые значения нажатием клавиши `<Enter>`; нажатием клавиши пробел.
7. Составьте программу, которая запрашивает у пользователя два целых числа и выводит квадрат суммы $(a+b)^2$ и сумму квадратов $a^2 + b^2$ этих чисел. Пример работы программы:

```
Введите два числа:
a=3
b=2
Квадрат суммы 3 и 2 равен 25
Сумма квадратов 3 и 2 равна 13
```

8. Составьте программу вычисления длины окружности $L=2*\pi*R$ и площади круга $S=\pi*R$ заданного радиуса R .
9. Даны два числа a и b . Найти их среднее арифметическое и среднее геометрическое.

Лабораторная работа № 2

Разработка, отладка и испытание разветвляющихся алгоритмов и программ

Цель работы: сформировать умения разрабатывать и отлаживать разветвляющиеся алгоритмы и программы.

Методические указания:

- изучить темы «Понятие алгоритма: свойства, способы описания», «Структура программного модуля. Состав интегрированной программной среды» и «Условный оператор, оператор выбора. Логические операции в Pascal»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое алгоритм?
2. Что такое программа разветвляющейся структуры?
3. Перечислите операторы, используемые в разветвляющихся программах.
4. Что такое метка и для чего она предназначена?
5. Запишите синтаксис оператора безусловного перехода.
6. Запишите синтаксис условного оператора (полная и неполная (краткая) форма).
7. Запишите синтаксис оператора выбора.
8. В каком случае необходимо заключать операторы в операторные скобки `begin ... end`?
9. Как записываются операции сравнения в языке Pascal?
10. Как записываются логические операции в языке Pascal?
11. Как правильно записать сложное условное выражение в языке Pascal?
12. Приведите пример задачи, алгоритм решения которой является разветвляющимся.

Задания к работе.

1. Выведите на экран большее из двух любых чисел, вводимых с клавиатуры.
2. Компьютер спрашивает Вашу отметку за четверть по математике и в зависимости от введенной отметки выводит: «Молодец!» (от 8 до 10) или «Надо учиться лучше» (от 0 до 7).
3. С клавиатуры вводится число. Если оно кратно трем, возведите его в квадрат, иначе разделите удвоенное произведение данного числа на три.
4. Составить программу, которая проверяет, является ли четным введенное целое число.
5. Вычислить значение функции:

$$y = \begin{cases} x^2 + 5, & \text{если } x > 3 \\ x - 8, & \text{если } x \leq 3 \end{cases}$$

6. Найти количество положительных чисел среди четырех целых чисел А, В, С и D.
7. Составить программу, которая запрашивает два числа, запрашивает ответ на сумму этих чисел, проверяет его и выводит сообщение «Правильно» или «Вы ошиблись» и правильный результат.
8. Найти действительные корни квадратного уравнения $ax^2 + bx + c = 0$, $a \neq 0$.
9. Составить программу для проверки знания даты начала Второй мировой войны. В случае неверного ответа пользователя программа должна вывести правильный ответ.

Исходные данные:

Результат:

1. Введите год начала Второй мировой войны: 1941.
2. Введите год начала Второй мировой войны: 1939.

1. Вы ошиблись, Вторая мировая война началась в 1939 году.
2. Верно.

10. Составить программу для определения стоимости разговора по телефону (25 руб. 1 мин) с учетом скидки 20%, предоставляемой по субботам и воскресеньям.

Исходные данные:

Результат:

1. Длительность разговора (целое число мин): 10
День недели (1 – понедельник, ..., 7 – воскресенье): 2
2. Длительность разговора (целое число мин): 4.
День недели (1 – понедельник, ..., 7 – воскресенье): 7.

1. Стоимость разговора: 250 руб.
2. Стоимость разговора: 80 руб.

11. В зависимости от стажа работы на предприятии введена надбавка в размере:

для работающих от 5 до 10 лет – 10%;
для работающих от 10 до 15 лет – 15%;
для работающих свыше 15 лет – 20%.

Составить программу, которая по заданному стажу работы определит размер надбавки в процентах (при решении задачи используйте оператор выбора).

12. В магазине Вам надо заплатить N рублей. Вы подаете продавцу m купюр по k рублей. Определите сколько денег Вам надо добавить или сколько Вам должны дать сдачи.

13. *Составьте программу, которая определяла бы вид треугольника по длинам его сторон a , b и c (если данные отрезки позволяют его построить). Условием того, что треугольник может быть составлен, является одновременное выполнение следующих условий: $a+b>c$, $b+c>a$, $a+c>b$, а также

то, что треугольники могут быть разными: равносторонними, равнобедренными, прямоугольными, равнобедренными прямоугольными и т. д. Кроме того, следует учесть, что в качестве длин сторон могут быть случайно введены как нулевые, так и отрицательные значения.

Лабораторная работа № 3

Разработка, отладка и испытание простых циклических алгоритмов и программ с заданным числом повторений

Цель работы: сформировать умения разрабатывать, отлаживать и испытывать простые циклические алгоритмы и программы с заданным числом повторений.

Методические указания:

- изучить темы «Понятие алгоритма: свойства, способы описания», «Структура программного модуля. Состав интегрированной программной среды», «Операторы организации циклов»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое цикл?
2. Какая программа называется программой циклической структуры?
3. Запишите синтаксис оператора цикла с параметром.
4. Что такое «параметр цикла»? Что такое «тело цикла»?
5. Какие типы данных могут быть использованы для счетчика цикла `for`?
6. В чем отличие между операторами `for..to` и `for..downto`?
7. Что требуется сделать для того, чтобы в теле цикла выполнялись несколько операторов?
8. Как найти, сколько раз выполняется цикл?
9. Приведите примеры задач, для решения которых используется структура цикл.
10. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 1 оператор. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 5 операторов.

Задания к работе.

1. Загрузите систему программирования PascalABC.Net. В появившемся окне введите следующий текст:

```
program fact;  
var n, i: word;  
    P: longint;
```

```

begin
write('n=');
readln(n);
P:=1;
for i:=2 to n do
P:=P*i;
writeln(n, '!=', P);
end.

```

Сохраните файл под именем FACT.PAS. Протестируйте содержащуюся в нем программу и проследите при этом за изменением переменных i и P . Выполните программу для $n=20$. Объясните полученный результат. Укажите диапазон значений n , для которых $n!$ программа вычисляет правильно. Измените программу так, чтобы ввод значения n повторялся до тех пор, пока не будет введено значение из указанного диапазона. Можно ли для этого использовать оператор цикла с постусловием? Оператор цикла с параметром?

- Измените программу FACT, полученную при выполнении задания 4, организовав вычисление произведения $1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ в обратном порядке, то есть $n \cdot n-1 \cdot \dots \cdot 2 \cdot 1$.
- Составьте программу вычисления суммы первых 15 членов ряда $1+2+3+\dots+n$.
- Найти сумму натуральных чисел от 2 до 56, кратных 5.
- Найти произведение натуральных нечетных чисел от n_1 до n_2 .
- В 1626 г. индейцы продали остров Манхэттен за 20 долларов. Если бы эти деньги были помещены в банк на текущий счет, и ежегодный прирост составлял бы 4%, какова была бы величина капитала в 2006 г.?
- Найти сумму положительных нечетных чисел, меньших 100.
- Дано натуральное число n ($n \leq 1999$). Определить, является ли оно палиндромом («перевертышем»), с учетом четырех цифр. Например, палиндромами являются числа: 2222, 6116, 0440.

Подсказка. Так как число четырехзначное, то переменная оператора For изменяется от 1 до 4. В переменной с именем m хранится «остаток» числа, в первоначальный момент времени он равен введенному числу. В переменной с именем r формируем значение числа – «перевертыша». Основными операциями являются: $r:=10*r + m \bmod 10$ (добавление очередной цифры к числу «перевертышу») и $m:=m \div 10$ (изменение проверяемого числа). Изменения переменных i , m и r см. в таблице.

i	m	r
–	3994	0
1	399	$0*10+3994 \bmod 10=0+4=4$
2	39	$4*10+399 \bmod 10=40+9=49$
3	3	$49*10+39 \bmod 10=490+9=499$
4	0	$499*10+3 \bmod 10=4990+3=4993$

- Измените программу, составленную при выполнении предыдущего задания так, чтобы можно было определять, является ли число палиндромом независимо от количества цифр в числе.

10. Вывести на экран все трехзначные числа, сумма цифр которых равна N . Попробовать перебрать все трехзначные числа от 100 до 999, затем каким-то образом делением и вычитанием выделять отдельные цифры, искать их сумму, сравнивать с N достаточно сложно. Исходя из того, что трехзначное число состоит из 3 цифр, причем первая изменяется от 1 до 9, а вторая и третья – от 0 до 9.

```
program primer10;
var i, j, k, n: integer;
begin
write('Введите N=');
readln(n);
for i:=1 to 9 do
for j:=0 to 9 do
for k:=0 to 9 do
if i+j+k=n then writeln(i, j, k);
end.
```

Замечание: оператор `writeln(i,j,k)` выводит на экран три цифры без пробела, поэтому они на экране выглядят как одно число. Хотя можно было вывести, учитывая, что i -сотни, j -десятки, k -единицы, и так: `writeln(i*100+j*10+k)`.

Лабораторная работа № 4

Разработка, отладка и испытание простых циклических алгоритмов и программ с неизвестным числом повторений

Цель работы: сформировать умения разрабатывать алгоритмы и программы итерационных циклов.

Методические указания:

- изучить темы «Понятие алгоритма: свойства, способы описания», «Структура программного модуля. Состав интегрированной программной среды», «Операторы организации циклов»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое цикл?
2. Какая программа называется программой циклической структуры?
3. Каков синтаксис оператора цикла с предусловием, с постусловием?
4. Что такое «тело цикла»?
5. Что требуется сделать для того, чтобы в теле цикла выполнялись несколько операторов?
6. Приведите примеры задач, для решения которых используется структура цикла с предусловием, с постусловием.

7. Что такое «итерация»?
8. Каков синтаксис оператора цикла с предусловием, с постусловием?
9. Что такое «условие выхода из цикла»?
10. Каким должно быть условие продолжения цикла с предусловием (истинным или ложным)?
11. Каким должно быть условие продолжения цикла с постусловием (истинным или ложным)?
12. Какие существуют операторы прерывания выполнения цикла? Охарактеризуйте каждый.

Задания к работе.

1. Определите значение переменной s после выполнения следующих операторов:
 $s:=0; i:=0; \text{ while } i<5 \text{ do } \text{inc}(i); s:=s+100 \text{ div } i;$
 $s:=0; i:=1; \text{ while } i>1 \text{ do } \text{begin } s:=s+100 \text{ div } i;$
 $\text{dec}(i); \text{end};$

Справка:

Dec(X,DX) уменьшает значение переменной X на величину DX , а если параметр DX не задан – на единицу. В данном случае выполнение процедуры $Dec(i)$ аналогично выполнению оператора $i:=i-1$.

Inc(X,DX) увеличивает значение переменной X на величину DX , а если параметр DX не задан – на единицу. В данном случае выполнение процедуры $Inc(i)$ аналогично выполнению оператора $i:=i+1$.

2. Составить программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *while*.
3. Составить программу нахождения суммы нечетных чисел последовательности от 0 до 100, с помощью цикла *repeat*.
4. Найти сумму чисел от 1, кратных 5. Складывать до тех пор, пока сумма не станет больше N .
5. Найти сумму $S=1/(2+7)+1/(5+7)+1/(8+7)+\dots$. Складывать до тех пор, пока слагаемое не станет меньше $1/N$.
6. Поменять порядок цифр числа на обратный. Например, было 12345, стало 54321.
7. Дано целое число $N (>0)$, являющейся некоторой степенью числа 2: $N=2^k$. Найти целое число k – показатель этой степени.
8. Начальный вклад в банке равен 1000 рублей. Через месяц размер вклада увеличивается на P процентов от имеющейся суммы (P – вещественное число, $0<P<25$). По данному P определить, через сколько месяцев размер вклада превысит 1100 рублей, и вывести найденное количество месяцев K (целое число) и итоговый размер вклада S (вещественное число).
9. *Сумма квадратов длин катетов a и b прямоугольного треугольника равна квадрату длины гипотенузы c : $a^2+b^2=c^2$. Тройка натуральных чисел, удовлетворяющих этому равенству, называется Пифагоровыми числами. Составить программу нахождения основных троек Пифагоровых чисел, ис-

пользуя следующие формулы: $a=u*v$; где u и v – взаимно простые нечетные натуральные числа и $u>v$.

Лабораторная работа № 5

Разработка, отладка и испытание простых циклических алгоритмов и программ с известным числом повторений обработки массивов

Цель работы: научить разрабатывать простые циклические алгоритмы и программы обработки массивов с известным числом повторений, осуществлять их отладку и испытание.

Методические указания:

- изучить тему «Массивы: определение, описание, размещение в памяти, использование»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое массив, элемент массива, индекс элемента массива?
2. Массив является структурированным типом данных?
3. Какого типа могут быть элементы массива?
4. Какого типа могут быть индексы элементов массива?
5. Что такое «размер» и «размерность» массива?
6. Как объявить одномерный целочисленный массив В из 20 элементов?
7. Какие существуют способы заполнения массива?
8. Какие бывают массивы?
9. Какой оператор цикла наиболее естественен для обработки массивов?
10. В чем отличия одномерного от двумерного массива?
11. Особенности использования функции random?

Задания к работе.

1. Загрузите систему программирования PascalABC.Net. В появившемся окне введите следующий текст:

```
program lab5_1;
const nn=100;
var a: array[1..nn] of real;
i,n:integer; s:real;
begin
  writeln('Введите количество элементов массива');
  readln(n);
  writeln('Введите элементы массива');
```

```

for i:=1 to n do
read(a[i]);
s:=0;
for i:=1 to n do
if a[i]>0 then s:=s+a[i];
writeln('Сумма =', s);
end.

```

Пояснения к программе:

```

program lab5_1; //Заголовок программы lab5_1
const nn=100; //Раздел объявления констант имя nn:
                значение 100
var //Раздел описания переменных a: array[1..nn] of real;
i,n:integer;
s:real;
//Массив a: размер nn элементов, тип - вещественный
//Параметр цикла: имя i, тип - целое число
//Размер массива: имя n, тип - целое число
//Сумма положительных чисел: имя s, тип - вещественное
число
begin //Раздел операторов (основной блок)
writeln('Введите количество элементов массива');
//Вывод на экран строки комментария 'Введите количество
элементов массива'
readln(n); //Считывание с клавиатуры целого числа и со-
хранение его под именем n
writeln('Введите элементы массива');
//Вывод на экран строки комментария 'Введите элементы
массива'
for i:=1 to n do Организация цикла с параметром по пере-
менной i, изменяющейся от 1 до n с шагом +1
read(a[i]); //Считывание элемента массива a с индексом i
s:=0; //Обнуление значения переменной s
for i := 1 to n do //Организация цикла с параметром по
переменной i, изменяющейся от 1 до n с шагом +1
if a[i]>0 then s:=s+a[i]; //Тело цикла: если элемент мас-
сива a с индексом i больше нуля, то увеличение суммы по-
ложительных чисел на значение элемента a[i]
writeln('Сумма =', s); //Вывод на экран строки «Сумма =»
и значения, хранимого под именем s
end. //Завершение раздела операторов (текста программы)

```

Формулировка задачи: найти сумму положительных элементов одномерного массива A размера N .

Модифицируйте программу так, чтобы она выполняла вычисление произведения положительных элементов массива и вычисление количества отрицательных элементов массива. Предусмотрите вывод элементов массива на экран в строку.

Сохраните программу в своей рабочей папке.

2. Одномерный массив заполнен целыми числами. Вычислить сумму его элементов, кратных 5.
3. Одномерный массив заполнен целыми числами. Возвести в квадрат элементы, стоящие на четных местах.
4. Одномерный массив заполнен целыми числами. Удвойте четные и утройте нечетные элементы.
5. Заполнить одномерный массив из 50 элементов случайными числами от -5 до 5. Организовать вывод массива в строку. Сколько получилось отрицательных чисел?
6. Заполнить массив случайными целыми числами из интервала от -7 до 25.
7. Написать программу, которая проверяет, есть ли во введенном с клавиатуры массиве элементы с одинаковыми значениями.
8. Найти произведение нечетных элементов двумерного массива целых чисел.
9. Найти сумму элементов двумерного массива, находящихся ниже главной диагонали.
10. Найти максимальный элемент дополнительной диагонали квадратной матрицы.

Лабораторная работа № 6

Разработка, отладка и испытание циклических алгоритмов и программ с известным числом повторений с внутренними ветвлениями

Цель работы: научить разрабатывать, отлаживать и испытывать циклические алгоритмы и программы с известным числом повторений с внутренними ветвлениями.

Методические указания:

- изучить темы «Условный оператор, оператор выбора. Логические операции в Pascal», «Операторы организации циклов», «Массивы: определение, описание, размещение в памяти, использование»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Назовите особенности операторов организации циклов. Запишите синтаксис каждого из цикла на языке Pascal.
2. Что такое «итерация»?
3. В каком случае в теле цикла необходимо использовать составной оператор?
4. Запишите синтаксис условного оператора (полная и неполная (краткая) форма).
5. Верно ли, что до начала итерационного цикла должны быть сделаны начальные установки переменных, управляющих условием цикла?
6. Как правильно записать сложное условное выражение в языке Pascal?

Задания к работе.

1. Найти сумму целых положительных чисел, больших 20, меньших 100, кратных 3 и заканчивающихся на 2, 4 или 8.
2. Написать программу поиска четырехзначного числа, которое при делении на 133 дает в остатке 125, а при делении на 134 дает в остатке 111.
3. Определить, является ли заданное число степенью 3.
4. Напишите программу, которая находит все четырехзначные числа $abcd$ (a, b, c, d – цифры числа, причем все они различны), для которых выполняется условие: $ab - cd = a + b + c + d$. Другими словами, разность чисел, составленных из старших цифр числа и из младших, равна сумме цифр числа.
5. Даны натуральные четырехзначные числа n, k ($n, k \leq 9999$). Из чисел от n до k выбрать те, запись которых содержит ровно три одинаковые цифры. Например, числа 6766, 5444, 0006, 0060 содержат ровно три одинаковые цифры.

6. Задан массив из N элементов. Определить среднее арифметическое наибольшего и наименьшего элементов.
7. Задан массив из M элементов. Определить сумму отрицательных элементов кратных 6.
8. Задан массив из N элементов. Определить сумму квадратов наибольшего и наименьшего элементов.

Лабораторная работа № 7

Разработка, отладка и испытание алгоритмов и программ с использованием процедур и функций

Цель работы: развить умения разрабатывать, отлаживать и испытывать алгоритмы и программы с использованием процедур и функций.

Методические указания:

- изучить темы «Процедуры и функции», «Операторы организации циклов», «Массивы: определение, описание, размещение в памяти, использование»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что называется подпрограммой?
2. Какова структура программы с подпрограммами?
3. Какова структура процедур?
4. Какова структура подпрограммы-функции?
5. Каково назначение формальных, фактических параметров?
6. В чем особенность параметров-значений, параметров-переменных?
7. Чем отличаются локальные и глобальные параметры? Какова область их действия?
8. Как обратиться к процедурам и функциям?
9. В чем состоит различие и сходство подпрограмм-процедур и подпрограмм-функций в Pascal?

Задания к работе.

1. Составьте программу для определения числа сочетаний $C_n^m = \frac{n!}{m!(n-m)!}$,

используя функцию при вычислении факториала:

```
function factor (n:integer): integer;
var i:integer; rez: integer;
begin
rez:=1;
```

```

for i:=1 to n do
rez:=rez*i;
factor:=rez;
end;

```

2. С помощью следующей процедуры произведите обмен значениями между двумя переменными:

```

procedure swap (var x,y: real);
var t:real; //промежуточная переменная
begin
t:=x; //запоминаем значение x
x:=y; //переменной x присваиваем значение
y
y:=t; //переменной y присваиваем t, равное
x
end;

```

3. Опишите функцию, определяющую максимальное из двух чисел, и с помощью этой функции вычислите значение выражения: $Max(A - B, A + B) + 2 * Max(A * B, A \bmod B)$.

4. Даны значения c, d, l, k . Напишите функцию \min для вычисления значения следующего выражения: $\frac{\min(c + d, l, k) + \min(k, c, l)}{3 + \min(k, d - l, c + l)}$.

5. Ввести 3 целых массива из n чисел каждый: A, B, C . Вычислить значение $r = \frac{\max(A) * \max(B)}{\max(C)}$, где $\max(A)$ – максимальный элемент массива A (аналогично $\max(B)$ и $\max(C)$).

Вычисление максимального элемента массива оформить в виде подпрограммы. Формула имеет смысл, если $\max(C) > 0$.

6. Треугольник задан длинами своих сторон. Найдите длины его медиан. Вычисление длины медианы оформить в виде подпрограммы ($m_b = \frac{1}{2} \sqrt{2a^2 + 2c^2 - b^2}$ – длина медианы, проведенной из вершины B).

Лабораторная работа № 8

Разработка алгоритмов и программ обработки строк

Цель работы: научить разрабатывать алгоритмы и программы обработки строк.

Методические указания:

- изучить тему «Символьные переменные и строки»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое символьная константа?
2. Каким идентификатором определяются данные строкового и символьного типов?
3. Описание строкового типа данных.
4. Какова максимальная длина строки?
5. Какие выражения называются строковыми?
6. Какие операции допустимы над строковыми данными?
7. Каким образом производится сравнение символов и строк?
8. Как можно обратиться к отдельным символам строки?
9. Каково назначение специальных процедур и функций обработки данных символьного типа?
10. Стандартные строковые процедуры и функции в языке Pascal.

Задания к работе.

1. Загрузите систему программирования PascalABC.Net. В появившемся окне введите следующий текст:

```

program alf;
var c:char;
begin
  for c:='A' to 'Z' do write(c, ' ');
  writeln;
end.

```

Выполните содержащуюся в нем программу. Измените программу так, чтобы на экран выводились прописные и строчные буквы алфавита следующим образом: Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz.

Сохраните файл под именем ALF.PAS в своей рабочей папке.

2. Измените программу, созданную при выполнении предыдущего задания, для вывода на экран прописных и строчных букв русского алфавита.
3. Из слова «ВЕЛИКОЛЕПНЫЙ» с помощью строковых функций получить слова: ВЕЛИК, КОЛ, ЛИК, ЛЕВ, ВЕК, вывести их на экран и определить их длину.
4. Дано слово. Получить слово, являющееся записью данного слова справа налево.
5. Дано длинное натуральное число. Используя процедуры `val` и `str` и/или функцию `length`, выполните следующее:
 - а) подсчитайте количество цифр в числе;
 - б) найдите произведение цифр числа и их сумму.
6. Задан текст. Удалите в тексте фрагмент, заключенный в круглые скобки.
7. Задан текст. Слова в тексте разделены пробелами. В конце текста – точка. Заменить второе слово в тексте на введенное слово.
8. Дано предложение, состоящее из слов, отделенных друг от друга одной звездочкой (*). Удалите из этого предложения слова длиной N символов.

Лабораторная работа № 9

Разработка алгоритмов и программ решения различных задач с использованием библиотечных модулей пользователя

Цель работы: сформировать умения разрабатывать алгоритмы и программы решения задач с использованием библиотечных модулей пользователя.

Методические указания:

- изучить тему «Организация библиотек. Стандартные библиотечные модули и модули пользователя. Структура Unit-а»;
- ответить на вопросы для подготовки к лабораторной работе;
- выполнить задания к работе.

Вопросы для подготовки к лабораторной работе:

1. Что такое модуль в программировании?
2. Для чего служат модули?
3. Назовите стандартные модули в Pascal.
4. Опишите структуру модуля.
5. Для чего служат ключевые слова unit, interface, implementation?
8. Как подключается модуль к разрабатываемой единице?
9. Где хранится разработанный модуль на диске?

Задания к работе.

1. Загрузите систему программирования PascalABC.Net. В появившемся окне введите следующий текст:

```
unit obmen;  
interface  
procedure swap(var a,b:integer);  
implementation  
procedure swap;  
var c:integer;  
begin  
c:=a;  
a:=b;  
b:=c;  
end;  
end.
```

Сохраните модуль под именем `obmen.pas` в своей рабочей папке. Данный модуль будет совершать обмен значений между двумя числами.

2. Создайте файл основной программы:

```
program modul_s;
```

```
uses obmen;
var a,b:integer;
begin
writeln('Введите a');
readln(a);
writeln('Введите b');
readln(b);
writeln('a=',a,' ', 'b=',b);
swap(a,b);
writeln('a=',a,' ', 'b=',b);
end.
```

Сохраните файл в своей рабочей папке. Модуль и основная программа должны находиться в одной и той же папке.

3. Разработать программу с подключением модуля для нахождения максимального значения целочисленного массива.
4. Разработать программу с подключением модуля для нахождения минимального значения целочисленного массива, а также суммы и среднего арифметического.
5. Разработать программу с подключением модуля для нахождения суммы элементов целочисленной матрицы, расположенных на побочной диагонали.
6. *Создать модуль для работы с комплексными числами: описать операции «сложение», «вычитание», «умножение», «деление» над комплексными числами $a+bi$ и $c+di$. Разработать программу для вычисления суммы, разности, произведения, частного чисел $1+2i$ и $9+5i$.

Глава III. КОНТРОЛЬ ЗНАНИЙ

Тема 1: ПОНЯТИЕ АЛГОРИТМА: СВОЙСТВА, СПОСОБЫ ОПИСАНИЯ

Тест 1.1

1. Алгоритм – это:
 - а) граф, указывающий порядок выполнения некоторого набора команд;
 - б) четкое описание последовательности действий, которые необходимо выполнить для получения результата;
 - в) правила выполнения определенных действий;
 - г) набор команд для компьютера.
2. Алгоритм называется линейным, если:
 - а) представим в табличной форме;
 - б) все операции выполняются последовательно одна за другой независимо от каких-либо условий;
 - в) составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
 - г) ход его выполнения зависит от истинности тех или иных условий.
3. Алгоритм называется циклическим, если:
 - а) составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
 - б) все операции выполняются последовательно одна за другой независимо от каких-либо условий;
 - в) ход его выполнения зависит от истинности тех или иных условий;
 - г) представим в табличной форме.
4. Алгоритм включает в себя ветвление, если:
 - а) составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
 - б) все операции выполняются последовательно одна за другой независимо от каких-либо условий;
 - в) ход его выполнения зависит от истинности тех или иных условий;
 - г) представим в табличной форме.
5. Свойством алгоритма является:
 - а) результативность;
 - б) линейность;
 - в) цикличность;
 - г) возможность выполнения алгоритма в обратном порядке.
6. Свойство алгоритма, заключающееся в том, что он должен представлять процесс решения задачи как последовательное исполнение простых (или ранее определенных) шагов (этапов), называется:
 - а) определенность;

- б) массовость;
 - в) понятность;
 - г) дискретность.
7. Свойство алгоритма, заключающееся в отсутствии ошибок, алгоритм должен быть составлен только из команд, понятных исполнителю, называется:
- а) определенность;
 - б) массовость;
 - в) понятность;
 - г) дискретность.
8. Свойство алгоритма, заключающиеся в том, что один и тот же алгоритм решения задачи производится в общем виде, т. е. его можно будет применять для некоторого класса задач, различающихся лишь исходными данными, называется:
- а) дискретность;
 - б) массовость;
 - в) результативность;
 - г) определенность.
9. Свойство алгоритма, заключающиеся в том, что алгоритм должен приводить к решению задачи за конечное число шагов, называется:
- а) дискретность;
 - б) определенность;
 - в) результативность;
 - г) массовость.
10. Свойство алгоритма, заключающиеся в том, что каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Это свойство обеспечивает выполнение алгоритма механически, не требуя никаких дополнительных указаний или сведений о решаемой задаче, называется:
- а) определенность;
 - б) массовость;
 - в) результативность;
 - г) дискретность.

Тест 1.2

1. Алгоритм, записанный на «понятном» компьютеру языке программирования, называется:
- а) программой;
 - б) исполнителем алгоритмов;
 - в) листингом;
 - г) протоколом алгоритма.
2. Свойство алгоритма – дискретность обозначает:
- а) разбиение алгоритма на конечное число простых шагов;

- б) каждая команда должна быть описана в расчете на конкретного исполнителя;
 - в) строгое движение вверх;
 - г) команды должны следовать последовательно друг за другом.
3. Программа – это...
- а) правила выполнения определенных действий;
 - б) ориентированный граф, указывающий порядок выполнения некоторого набора команд;
 - в) это последовательность команд компьютера, приводящая к решению задачи;
 - г) набор команд для компьютера.
4. Как называется графическое представление алгоритма:
- а) таблица;
 - б) псевдокоды;
 - в) словесное описание;
 - г) блок-схема.
5. Графическое задание алгоритма – это:
- а) способ представления алгоритма с помощью геометрических фигур;
 - б) схематическое изображение в произвольной форме;
 - в) представление алгоритма в форме таблиц и расчетных формул;
 - г) система обозначения правил для единообразной и точной записи алгоритмов их исполнения.
6. Какой из документов является алгоритмом?
- а) правила техники безопасности;
 - б) технология по приготовлению торта;
 - в) список книг в библиотеке;
 - г) расписание автобусов.
7. Какой алгоритм должен быть выбран при решении квадратного уравнения?
- а) линейный;
 - б) разветвляющийся;
 - в) циклический;
 - г) линейно-циклический.
8. На кого рассчитан алгоритм, написанный на естественном языке?
- а) на робота;
 - б) на компьютер;
 - в) на человека;
 - г) на всех одновременно.
9. Какая фигура в блок-схеме предназначена для вывода данных?
- а) овал;
 - б) прямоугольник;
 - в) трапеция;
 - г) параллелограмм.
10. Какая фигура в блок-схеме предназначена для решения или действия?

- а) овал;
- б) прямоугольник;
- в) трапеция;
- г) параллелограмм.

Тема 2: СТРУКТУРА ПРОГРАММНОГО МОДУЛЯ. СОСТАВ ИНТЕГРИРОВАННОЙ ПРОГРАММНОЙ СРЕДЫ

Тест 2.1

1. Разработчиком языка Паскаль является:
 - а) Эдсгер В. Дейкстра;
 - б) Блез Паскаль;
 - в) Никлаус Вирт;
 - г) Норберт Винер.
2. Что из указанного не входит в алфавит языка Паскаль?
 - а) служебные слова;
 - б) латинские строчные и прописные буквы;
 - в) русские строчные и прописные буквы;
 - г) знаки арифметических действий.
3. Какая последовательность символов не может служить именем в языке Паскаль?
 - а) kaP1;
 - б) 2t;
 - в) t2;
 - г) mus.
4. В каком разделе происходит описание переменных?
 - а) `var`;
 - б) `var`;
 - в) `const`;
 - г) `begin`.
5. Символьный тип данных это – ...
 - а) `string`;
 - б) `boolean`;
 - в) `char`;
 - г) `integer`.
6. Целочисленные данные относятся к типу...
 - а) `real`;
 - б) `boolean`;
 - в) `integer`;
 - г) `string`.
7. Вещественные данные относятся к типу...

- a) boolean;
 - б) real;
 - в) integer;
 - г) char.
8. Чем характеризуется переменная?
- а) значением, типом;
 - б) именем, типом, значением;
 - в) именем, значением;
 - г) значением.
9. Как записывается оператор вывода?
- а) writeln;
 - б) readkey();
 - в) readln();
 - г) readwrit.
10. Как записывается оператор ввода?
- а) writeln;
 - б) readwrit;
 - в) readln();
 - г) readkey().

Тест 2.2

1. Оператор для организации диалога с пользователем в языках программирования – это...
- а) условный оператор;
 - б) оператор ввода и оператор вывода;
 - в) оператор цикла;
 - г) оператор выбора.
2. Как записывается оператор присвоения?
- а) b:=2;
 - б) b=2;
 - в) b=:2;
 - г) :=2.
3. Какая строка описывает вещественную переменную на языке Паскаль?
- а) var x: boolean;
 - б) var x: integer;
 - в) var x: real;
 - г) var x: char.
4. Какая строка описывает логическую переменную на языке Паскаль?
- а) var x: boolean;
 - б) var x: integer;
 - в) var x: char;
 - г) var x: real.
5. Описать переменную – это значит указать ее...

- а) имя, тип и значение;
 - б) имя и тип;
 - в) тип и значение;
 - г) имя и значение.
6. При присваивании изменяется ...
- а) имя переменной;
 - б) тип переменной;
 - в) значение константы;
 - г) значение переменной.
7. Как будут выведены значения: `writeln(a); writeln(b);`?
- а) в столбик;
 - б) через пробел;
 - в) через точку с запятой;
 - г) через 10 позиций.
8. Какого раздела не существует в программе, написанном на языке Паскаль?
- а) описаний;
 - б) заголовков;
 - в) операторов;
 - г) примечаний.
9. Разделителями между операторами служит...
- а) пробел;
 - б) запятая;
 - в) точка с запятой;
 - г) точка.
10. Определите значения переменных `s` и `i` после выполнения фрагмента программы:
- ```
s:=0; i:=5;
while i>0 do
begin
s:=s+i;
i:=i-1;
```
- а) `s=0, i= -1`;
  - б) `s=15, i=0`;
  - в) `s=5, i=0`;
  - г) `s=15, i=5`.

### Тест 2.3

1. Какая строка описывает символьную переменную на языке Паскаль?
- а) `var x: boolean;`
  - б) `var x: char;`
  - в) `var x: real;`
  - г) `var x: string.`
2. В данном фрагменте программы сделана ошибка:

```
program error;
begin
das:=25-14;
end.
```

- а) некорректное имя программы;
  - б) некорректное имя переменной;
  - в) запись арифметического выражения;
  - г) не определено имя переменной.
3. Какую клавишу следует нажать после набора последнего данного в операторе read?
- а) точка с запятой;
  - б) пробел;
  - в) enter;
  - г) delete.
4. Символ-разделитель операторов в Паскале ...
- а) запятая;
  - б) пробел;
  - в) точка;
  - г) точка с запятой.
5. Для вычисления квадратного корня из  $x$  используется функция:
- а) `sqrt(x)`;
  - б) `abs(x)`;
  - в) `sq(x)`;
  - г) `int(x)`.
6. Оператор присваивания в Паскале ...
- а) заносит в память и запоминает значение любой величины;
  - б) обозначается как `(:=)` и предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, идентификатор которой расположен в левой части;
  - в) вычисляет значение величины, указанной в левой части оператора;
  - г) отождествляет значения нескольких переменных значению, указанному в правой части оператора.
7. Для чего предназначен оператор `writeln` и `write`?
- а) для ввода данных с клавиатуры;
  - б) для ввода числовых данных;
  - в) для печати результатов на принтере;
  - г) для вывода на экран.
8. Значения переменных  $a$  и  $b$  после выполнения фрагмента программы:
- ```
a:=1;  
b:=10;  
a:=a+b;  
b:=a-b;  
a:=a-b;
```
- а) останутся прежними;

- б) поменяются местами;
 - в) станут равными соответственно сумме прежних своих значений;
 - г) станут равными соответственно разности прежних своих значений.
9. Какое значение примет переменная A после выполнения оператора $A = 17 \bmod 4$?
- а) 4.25;
 - б) 1;
 - в) 5;
 - г) 4.
10. Какое значение примет переменная A после выполнения оператора $A := 15 \operatorname{div} 4$?
- а) 3;
 - б) 3.5
 - в) 0;
 - г) 4.

Тест 2.4

1. Какая процедура обеспечивает ввод данных в Паскале?
- а) `vegin`;
 - б) `writeln`;
 - в) `read` или `readln`;
 - г) `print`.
2. Какая процедура обеспечивает вывод данных в Паскале?
- а) `vegin`;
 - б) `writeln` или `write`;
 - в) `read` или `readln`;
 - г) `print` или `println`.
3. Какое значение будет иметь переменная A в выражении $A := 16/2 - (9+1)$;
- а) -2;
 - б) 1;
 - в) -1;
 - г) 0.
4. Определите значение переменной c после выполнения следующего фрагмента программы:
- ```
a:= 100;
b:= 30;
a:= a-b*3;
if a > b then c:= a-b else c:= b-a;
```
- а) -10;
  - б) 50;
  - в) 20;
  - г) 100.
5. Какое из перечисленных значений может быть вещественным?
- а) средний балл;

- б) количество книг в библиотеке;
  - в) количество месяцев в году;
  - г) количество курсов в колледже.
6. Какое из перечисленных значений может быть целочисленным?
- а) средний балл;
  - б) значение числа  $\pi$ ;
  - в) значение  $\sqrt{5}$ ;
  - г) количество учащихся в колледже.
7. Какое из арифметических действий будет выполняться в выражении без скобок первым: сложение или вычитание?
- а) сложение;
  - б) вычитание;
  - в) в порядке очередности;
  - г) сложение и вычитание одновременно.
8. Чему равно значение функции  $\text{ABS}(\text{SQRT}(169))$ ?
- а) 0;
  - б) 1;
  - в) 13;
  - г) 169.
9. Тип переменной, определяющий количество предметов на 2 курсе?
- а) целый;
  - б) символьный;
  - в) вещественный;
  - г) логический.

### Тест 2.5

1. Что вычисляет функция  $\text{mod}$ ?
- а) квадратный корень;
  - б) модуль числа;
  - в) остаток от деления;
  - г) определяет знак числа.
2. Что вычисляет функция  $\text{abs}$ ?
- а) квадратный корень;
  - б) модуль числа;
  - в) остаток от деления;
  - г) определяет знак числа.
3. Что вычисляет функция  $\text{sqrt}$ ?
- а) квадратный корень;
  - б) модуль числа;
  - в) остаток от деления;
  - г) определяет знак числа.
4. Чему равно значение выражения:  $-\text{abs}(-\text{sqrt}(225))$ ?
- а) 15;

- б) 225;  
в) -15;  
г) 1.
5. Чему равно значение выражения  $\sqrt{\text{abs}(-64)}$ ?
- а) -64;  
б) 64;  
в) -8;  
г) 8.
6. Определите результат работы программы: `writeln('1+1=',1+1);`
- а) `1+1=1+1;`  
б) `2=2;`  
в) `4;`  
г) `1+1=2.`
7. Определите результат работы программы:
- ```
A:=-2;  
B:=-3;  
A:=B;  
B:=A;  
writeln (A,B);
```
- а) -2,-3;
б) -3,-3;
в) -3,-2;
г) 0,0.
8. Значением выражения $20 \text{ div } 6$ будет...
- а) 2;
б) 3;
в) 4;
г) 14.
9. Значением выражения $2 \text{ mod } 5$ будет...
- а) 0;
б) 1;
в) 2;
г) 3.
10. Какое значение примет переменная *a* в результате выполнения фрагмента программы:
- ```
a:=4;
b:=a*2+1;
a:=a+b;?
```
- а) 12;  
б) 13;  
в) 14;  
г) 20.

### Задание 2.1

1. Вычислите значение выражений там, где это возможно:

- а)  $25 \operatorname{div} 6$ ;
- б)  $25 \operatorname{mod} 6$ ;
- в)  $500 \operatorname{div} 10$ ;
- г)  $500 \operatorname{mod} 100$ ;
- д)  $6 \operatorname{div} 8$ ;
- е)  $17 \operatorname{mod} 5$ ;
- ж)  $140 \operatorname{div} 0$ ;
- з)  $6.0 \operatorname{mod} 2$ .

2. Запишите числа в форме с фиксированной точкой:

- а)  $5.3\text{E}+07$ ;
- б)  $-5.5\text{E}+04$ ;
- в)  $2.7\text{E}+00$ ;
- г)  $7.2\text{E}-05$ .

3. Записать на Паскале следующие выражения:

- а)  $x^4$ ;
- б)  $x^{-4}$ ;
- в)  $x^{\sqrt{2}}$ ;
- г)  $x^{x^2}$ ;
- д)  $x^2 + \sqrt{x}$ .

4. Записать на языке Паскаль следующие выражения:

- а)  $y = a \div 2b$
- б)  $y = v \cdot \frac{1}{2}b$
- в)  $y = \frac{a + 2b}{5s}$
- г)  $y = \frac{a - 8b}{57r}$
- д)  $y = \frac{a}{bc} + \frac{d}{rt}$
- е)  $y = \frac{k}{c} + \frac{e}{r+t}$
- ж)  $y = ae + nu$
- з)  $y = fh + qw$
- и)  $y = x^2 + dx + k$
- к)  $y = ax^2 + zx + p$
- л)  $y = \frac{a^2}{c} + \frac{d\sqrt{t+u}}{r^2t}$
- м)  $y = \frac{b^2}{t} - \frac{p\sqrt{t-2u}}{av^2}$

$$\text{н) } y = \sqrt{\frac{d\sqrt{t+u}}{r^2t}}$$

$$\text{о) } y = \sqrt{-\frac{p\sqrt{t-2u}}{av^2}}$$

$$\text{п) } y = e^{\sqrt{\frac{d\sqrt{t+u}}{r^2t}}}$$

$$\text{р) } y = e^{\sqrt{-\frac{p\sqrt{t-2u}}{av^2}}}$$

$$\text{с) } y = \frac{a \cdot 2,5 \cdot e^x + \sqrt{|x-1|}}{\sqrt{a+b^2} - \sqrt{c-d^2}}$$

$$\text{т) } z = \frac{b : 2 \cdot e^x - \sqrt{|x-1+y|}}{\sqrt{a+b^2} + \sqrt{m^2+n^2}}$$

5. Вычислить выражение:

- а)  $abs(18/(6+3) - 18/3/6 - 18/3 * 6)$ ;
- б)  $sqr(sqr(36) + trunc(1.2))$ ;
- в)  $5 * 7 div 4 mod 5 / 3 - trunc(5.9)$ .

### **Тема 3: УСЛОВНЫЙ ОПЕРАТОР, ОПЕРАТОР ВЫБОРА. ЛОГИЧЕСКИЕ ОПЕРАЦИИ В PASCAL**

#### **Тест 3.1**

1. Когда используется выражение, в котором есть слово then, то в выражении обязательно должно быть слово ...
  - а) input;
  - б) else;
  - в) if;
  - г) goto.
2. Какие два условных оператора имеются в языке Паскаль?
  - а) if <условие> then «оператор» или if <условие> then «оператор1» else «оператор2»;
  - б) операторы: if и case;
  - в) краткая и полная формы условного оператора if;
  - г) оператор выполнения цикла по условию и оператор if.
3. Формат полного оператора ветвления имеет вид ...
  - а) if <логическое выражение> else <оператор2>;
  - б) if <логическое выражение> then <оператор1> else <оператор2>;
  - в) if <логическое выражение> then <оператор1>;

- г) if then <оператор1> else <оператор2>.
4. Когда вы используете выражение, которое начинается со слова if, то обязательно применяете слово:
- else;
  - than;
  - then;
  - goto.
5. В каких операторах допущены ошибки?
- if a> <b then c:=a-b;
  - if a>b c:=a+b else c:=a-b;
  - if a>b then c:=a-b else c:=a+b;
  - if a>b c:=a+b d:=a\*b else c:=a-b;
6. Имеется условный оператор:  
if D< >10 then writeln ('хорошо') else writeln ('плохо');  
Можно ли заменить его следующими операторами?
- if D< >10 then writeln ('плохо') else writeln ('хорошо');
  - if not (D=10) then writeln ('плохо') else writeln ('хорошо');
  - if not (D< >10) then writeln ('плохо') else writeln ('хорошо');
  - if D=10 then writeln ('хорошо') else writeln ('плохо');
7. Укажите условие выбора чисел, кратных 3, но не кратных 6:
- (a mod 3=0);
  - (a mod 3=0) and (a mod 6<>0);
  - (a mod 3=0) or (a mod 6<>0);
  - not ((a mod 3=0) and (a mod 6<>0)).
8. Есть ли ошибка в использовании знака присвоения «:=» в представленном фрагменте программы:  
x:=100;  
if x=100 then y:=x\*x; ?
- неправильно использован знак присвоения «:=» в операторе «if x=100 then y:=x\*x;» после слова if надо писать не «x=100», а «x:=100»;
  - ошибок нет;
  - в операторе «if x=100 then y:=x\*x;» после слова then необходимо писать «y=x\*x» вместо «y:=x\*x»;
  - необходимо писать «x=100; if x:=100 then y:=x\*x;»
9. Могут ли операторы if быть вложенными?
- да, могут; в этом случае последнее «else» относится к ближайшему незанятому «if»;
  - нет, не могут;
  - всегда могут;
  - не знаю.
10. Выберите оператор, определяющий, является ли число *a* четным.
- if a mod 2 = 0 then write ('Да') else ('Нет');
  - if a mod 2 = 1 then write ('Да') else ('Нет');



- в) if  $a \bmod 2 = 0$  then write ('Нет') else ('Да');
- г) if  $a \bmod 2 = 1$  then write ('Нет') else ('Да').

### Тест 3.2

1. Повествовательное предложение, в котором что-либо утверждается или отрицается, называется ...
  - а) высказыванием;
  - б) повествованием;
  - в) умозаключением;
  - г) анализом.
2. Какое значение примет переменная  $c$  в результате выполнения программы:  
var a, b, c: integer;  
begin  
a:=4;  
b:=a\*3-6;  
if b>2\*a then a:=2;  
if b<2\*a then a:=5;  
if b=2\*a then a:=b-a;  
c:=a\*b+a-b;  
writeln ('c=', c);  
end.
  - а) 8;
  - б) 28;
  - в) 22;
  - г) 29.
3. В каком из условных операторов допущена ошибка?
  - а) if  $a > b$  then max:=a else max:=b;
  - б) if  $a < b$  then min:=a; else min:=b;
  - в) if  $b = 0$  then writeln ('Деление невозможно');
  - г) if  $(a > b)$  and  $(b > 0)$  then c:=a+b.
4. Операция, связывающая два простых логических выражения и содержащая конструкцию «если – то», называется...
  - а) конъюнкцией;
  - б) дизъюнкцией;
  - в) импликацией;
  - г) отрицанием.
5. Условный оператор  
if  $a \bmod 2 = 0$  then  
write ('Да')  
else ('Нет');  
позволяет определить, является ли число  $a$  ...
  - а) целым;
  - б) четным;

- в) двузначным;  
г) простым.
6. Конъюнкция (логическое умножение) – соединение двух логических выражений (высказываний) с помощью союза ...
- а) «и»;  
б) «или»;  
в) «не»;  
г) «может».
7. Какие значения примут переменные  $s$  и  $d$  в результате выполнения программы:
- ```

Program vvv2;
var a, b, c, d: integer;
begin
a:=7;
b:=2*a-3;
if b>a then c:=b-a
else
d:=a-b;
writeln ('c=', c);
writeln ('d=', d);
end.

```
- а) $c=18, d=4$;
б) $c=18, d=4$;
в) $c=14, d=-14$;
г) $c=0, d=-14$.
8. Дизъюнкция (логическое сложение) – соединение двух логических высказываний с помощью союза ...
- а) «и»;
б) «или»;
в) «не»;
г) «может».
9. Какое значение примет переменная k в результате выполнения фрагмента программы:
- ```

k:=8;
while k<20 do k:=k+10;

```
- а) 20;  
б) 26;  
в) 28;  
г) 32.
10. Сколько раз повторится тело цикла в процессе выполнения фрагмента программы:
- ```

k:= 8;
while k<20 do k:=k+10;

```
- а) 0;

- б) 2;
- в) 4;
- г) 10.

11. Отрицание (инверсия) – добавляется частица ...

- а) «и»;
- б) «или»;
- в) «не»;
- г) «по».

Задание 3.1

1. Среди следующих выражений выберите логические:

- а) $(x=3) \text{ or } (x<>4)$;
- б) $2*x+5$;
- в) $x \bmod 2 = 0$;
- г) $x \text{ div } 5$;
- д) $x+y=10$;
- е) $x+y$;
- ж) $(x+y<2) \text{ and } x$;
- з) $(x>3) \text{ and } (x<10)$.

2. Составьте логические выражения:

- а) число a делится на 3 и не делится на 5;
- б) число a не делится на 3;
- в) число a делится на 6;
- г) число a меньше 0, но больше -9;
- д) число a больше 3, но меньше 1;
- е) число a положительное, кратное 3;
- ж) число a кратное 15.

Тема 4: ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ

Тест 4.1

1. Назначение циклической структуры:

- а) проверка условия в тексте;
- б) повторение одной команды не более 10 раз;
- в) повторение идущих подряд одинаковых команд некоторое число раз;
- г) выполнение идущих подряд одинаковых команд.

2. Цикл for называется ...

- а) цикл с предусловием;
- б) цикл с параметром;
- в) цикл с постусловием;
- г) цикл с ветвлением.

3. Тип переменных для параметра цикла for ...
 - а) логический;
 - б) целочисленный;
 - в) вещественный;
 - г) символьный.
4. Какой из перечисленных операторов цикла является оператором цикла с предусловием?
 - а) loop;
 - б) for;
 - в) while;
 - г) repeat ... until.
5. Цикл repeat называется ...
 - а) цикл с предусловием;
 - б) цикл с параметром;
 - в) цикл с постусловием;
 - г) цикл с ветвлением.
6. Какой из перечисленных операторов цикла является оператором цикла со счетчиком?
 - а) loop;
 - б) for;
 - в) while;
 - г) repeat ... until.
7. Какого оператора цикла не существует в Паскале?
 - а) loop;
 - б) for;
 - в) while;
 - г) repeat ... until.
8. Какой из перечисленных операторов цикла является оператором цикла с постусловием?
 - а) while;
 - б) repeat ... until;
 - в) for;
 - г) if ...then ... else.
9. Оператор цикла с предусловием в Паскале имеет следующий формат ...
 - а) write<выражение> do <оператор>;
 - б) writeln<выражение> go <оператор>;
 - в) while <выражение> do <оператор>;
 - г) while <оператор> do <выражение>.
10. В каком из операторов допущена синтаксическая ошибка?
 - а) for i=1 to 20 do p:=p+1;
 - б) while s<3 do s:=s-3;
 - в) for i:=10 downto 5 do p:=p+1;
 - г) repeat k:=k+1 until k<7.

Тест 4.2

1. Сколько строк напечатает программа:
var k,l: integer;
begin
for k:=8 downto 1 do
for l:=10 to 14 do
writeln('Язык Паскаль');
end.
а) 8;
б) 14;
в) 10;
г) 40.
2. После выполнения фрагмента программы
C:=0; A:=17; B:=13; D:=2*A+3;
while D>=B do begin
C:=C+1;
D:=D-B;
end;
значения переменных C и D равны...
а) C=2,D=11;
б) C=1,D=24;
в) C=0,D=37;
г) C=5,D=-12.
3. После выполнения фрагмента программы
C:=0; A:=100; D:=11; B:=A/2+3;
while D<=B do begin
C:=C+1;
B:=B-D;
end;
значения переменных C и B равны...
а) C=9,B=4;
б) C=3,B=20;
в) C=4,B=9;
г) C=1,B=9.
4. После выполнения фрагмента программы
K:=3;
for i:=1 to 4 do
K:=6*i-2*K;
переменная K примет значение...
а) 0;
б) 12;
в) -6;
г) 36.

5. После выполнения фрагмента программы
`S:=2;`
`for i:=1 to 4 do`
`S:=S*i+2*i;`
 переменная S примет значение...
- 4;
 - 176;
 - 42;
 - 12.
6. После выполнения фрагмента программы
`y:=0; x:=1;`
`repeat`
`y:=y+sqr(x);`
`x:=x+1`
`until x>=4;`
 переменная y примет значение...
- 2;
 - 14;
 - 5;
 - 1.
7. После выполнения фрагмента программы
`a:=0; b:=1;`
`repeat`
`a:=sqr(b)-a;`
`b:=b+1`
`until b=5;`
 переменная a примет значение...
- 10;
 - 3;
 - 5;
 - 1.
8. Сколько раз выполняется цикл?
`k := 100; for k := k to k do;`
- 0;
 - 1;
 - 100;
 - произойдет заикливание.
9. Чему равно значение переменной i в приведенном ниже коде?
`var l, i : integer;`
`begin`
`i:=0;`
`for l:=1 to 10 do`
`inc(i);`
`for l:=1 to 10 do`

```
inc(i);
for l:=1 to 10 do
inc(i);
writeln(i)
end.
```

- a) 30;
- б) 10;
- в) 20;
- г) 12.

10. Чему равно значение переменной j в приведенном ниже коде?

```
var l, j : integer;
begin
j:=0;
for l:=1 to 10 do
for l:=1 to 10 do
for l:=1 to 10 do
inc(j);
writeln(j)
end;
```

- a) 30;
- б) 10;
- в) 20;
- г) произойдет заикливание.

Тест 4.3

1. В данном фрагменте программы

```
s := 0;
for i := 1 to 10 do s := s+2*i;
вычисляется ...
```

- a) сумма первых десяти четных чисел;
- б) сумма четных чисел от 1 до 10;
- в) сумма целых чисел от 1 до 10;
- г) удвоенная сумма целых чисел от 1 до 10.

2. Определите значение переменной S после выполнения операторов: i:=0;

```
S:=0
while i<3 do
begin
i:=i+1;
S:=S+sqr(i);
end;
```

- a) 0;
- б) 11;
- в) 14;
- г) 18.

3. Определите значение переменной f после выполнения операторов:
- ```

x:=-24;
y:=60;
m:=abs(x);
n:=abs(y);
while (m > 0) and (n > 0) do
begin
if m > n then m :=2*(m mod n)
else n :=2*(n mod m)
end;
f:=m+n;

```
- а) 20;  
б) 24;  
в) 22;  
г) 26.
4. Вычисление  $S=5!$  можно описать следующим образом:
- а)  $S:=1; i:=2; \text{while } i \leq 5 \text{ do begin } S:=S*i; i:=i+1 \text{ end};$   
б)  $S:=1; i:=2; \text{repeat } S:=S*i; i:=i+1 \text{ until } i < 5;$   
в)  $S:=1; i:=2; \text{repeat } S:=S*i; i:=i+1 \text{ until } i > 5;$   
г)  $S:=1; i:=2; \text{while } i \leq 5 \text{ do } S:=S*i; i:=i+1.$
5. Что напечатает следующая программа?
- ```

var x: real;
label 10, 20, 30, 40, 50, 60, 70;
begin
x:=4; goto 50;
10: x:=sqr(x); goto 40;
20: writeln(x); goto 70;
30: x:=x-1; goto 10;
40: x:=x*2+2; goto 60;
50: x:=x+x; goto 30;
60: x:=sqrt(x); goto 20;
70: end.

```
- а) 16;
б) 100;
в) 10;
г) 25.
6. Чему равно значение переменной j в приведенном ниже коде?
- ```

var l, j: integer;
begin
j:=0;
for l:=1 to 10 do
for l:=1 to 10 do
for l:=1 to 10 do
inc(j);

```



```
inc(j);
inc(j);
writeln(j);
end.
```

- a) 30;
- б) 10;
- в) 20;
- г) 12.

7. Какого значение переменной S после выполнения операторов:

```
S:=0;
i:=1;
repeat S:=S+1/i;
i:=i-1
until i<=1;
```

- a) 2;
- б) 0;
- в) 1;
- г) произойдет зацикливание.

8. Что напечатает следующая программа?

```
var y : real;
label 1, 2, 3, 4, 5, 6, 7;
begin { какова программа? }
y:=5; goto 5;
1: y:=y/8; goto 4;
2: writeln(y); goto 7;
3: y:=y-1; goto 1;
4: y:=y*5+1; goto 6;
5: y:=sqr(y); goto 3;
6: y:=sqrt(y); goto 2;
7: end.
```

- a) 16;
- б) 4;
- в) 3;
- г) 25.

9. Какое количество раз выполнится цикл в представленном фрагменте программы:

```
p:=2;
repeat
p:=p*0.1
until p < 0.1;?
```

- a) 0 раз;
- б) 1 раз;
- в) 2 раза;
- г) произойдет зацикливание.

10. Какое количество раз выполнится цикл в представленном фрагменте программы:

```
a:=1;
b:=1;
while a+b <8 do
begin
a:=a+1;
b:=b+2
end;
```

- а) бесконечное число раз;
- б) 2 раза;
- в) 1 раз;
- г) не выполнится ни разу.

#### Задание 4.1

1. После выполнения фрагмента программы

```
n:=5;
S:=0;
i:=1;
while i<=n do begin
S:=S+i;
i:=i+1;
end;
```

конечная сумма  $S$  будет равна...

- а) В задании 1 вместо цикла **while** организуйте цикл с **постусловием**. Результат запишите.
- б) В задании 1 вместо цикла **while** организуйте цикл с **параметром**. Результат запишите.

2. После выполнения фрагмента программы

```
n:=6;
S:=0;
i:=10;
repeat
S:=S+i;
i:=i-1;
until i<n;
```

конечная сумма  $S$  будет равна...

- а) В задании 2 вместо цикла **repeat** организуйте цикл с **предусловием**. Результат запишите.
- б) В задании 2 вместо цикла **repeat** организуйте цикл с **параметром**. Результат запишите.

## Тема 5: МАССИВЫ: ОПРЕДЕЛЕНИЕ, ОПИСАНИЕ, РАЗМЕЩЕНИЕ В ПАМЯТИ, ИСПОЛЬЗОВАНИЕ

### Тест 5.1

1. Чем является program n\_3?
  - а) блок описания используемых данных;
  - б) программным блоком;
  - в) заголовок программы.
2. Что такое массив?
  - а) упорядоченное множество однотипных элементов, которым присваивается общее имя, различающихся индексами;
  - б) это вспомогательный объект, делающий более удобной работу пользователя при вводе, просмотре данных;
  - в) выражение, задающее некоторую последовательность действий по преобразованию данных.
3. Что такое индекс в одномерном массиве?
  - а) порядковый номер элемента массива;
  - б) наибольший размер элемента массива;
  - в) имя массива;
  - г) размерность массива.
4. Для заполнения массива путем ввода чисел с клавиатуры используется оператор:
  - а) write, writeln;
  - б) until;
  - в) random;
  - г) read, readln.
5. Для чего производится описание массивов?
  - а) чтобы пользователю запомнить, сколько ячеек в массиве;
  - б) чтобы компьютер запомнил имя массива;
  - в) чтобы компьютер зарезервировал память для хранения элементов массива;
  - г) чтобы компьютер зарезервировал количество энергии для обработки массива.
6. Номер элемента двумерного массива определяется:
  - а) порядковым номером элемента в линейной таблице;
  - б) номером строки элемента в таблице;
  - в) пересечением строки и столбца элемента в таблице;
  - г) пересечением строки и столбца элемента в таблице.
7. Укажите правильное описание двумерного массива:
  - а) a: array [1..n, 1..m] of integer;
  - б) a: aray [1..n,1..n] of integer;
  - в) a: array[1..m] of array [1..m] of integer;
  - г) a: array {1..n,1..m} of integer.

8. Выберите правильное описание массива A, состоящего из нескольких переменных целого типа:
- A: array [1..25] of integer;
  - A: array [1..25] of real;
  - A: array [1..10] integer;
  - A: [1..25] of integer.
9. Массив описан следующим образом:  
`const b: array [1..5] of integer=(1,2,3,5,4);`  
 Значение выражения  $b[5]/b[2]+b[1]-b[3]-b[4]$  равно:
- 22;
  - 5;
  - 0;
  - 1.
10. Какие значения примут элементы массива A[3] и A[4] после выполнения последовательности операторов, если первоначально  $A[3]:=5$ ;  
 $A[4]:=6$ ;  
 $V:=A[3]$ ;  
 $A[3]:=A[4]$ ;  
 $A[4]:=V$ ?
- $A[3]=6$ ;  $A[4]=5$ ;
  - $A[3]=5$ ;  $A[4]=6$ ;
  - $A[3]=6$ ;  $A[4]=6$ ;
  - $A[3]=5$ ;  $A[4]=5$ .

### Тест 5.2

1. Числовой одномерный массив A заполнен последовательно числами 4, 8, 13, 25. Укажите значение элемента A[2].
- 12;
  - 8;
  - 25;
  - 13.
2. Выберите ответ, соответствующий следующей записи:  $A[8]:=18$ .
- значение массива A равно 18;
  - индекс элемента массива A равен 8;
  - значение элемента массива A равно 8;
  - в массиве A 18 элементов.
3. Какие значения примут элементы массива A[3] и A[4] после выполнения последовательности операторов, если первоначально  $A[3]:=5$ ;  
 $A[4]:=6$ ;  
 $A[3]:=A[4]$ ;  
 $A[4]:=A[3]$ ?
- $A[3]=6$ ;  $A[4]=6$ ;
  - $A[3]=6$ ;  $A[4]=5$ ;

- в)  $A[3]=5; A[4]=6;$   
 г)  $A[3]=5; A[4]=5.$
4. Заполнение массива:  
 for i:=1 to 5 do a[i]:=sqr(i). Чему равно  $a[2]+a[5]$ ?  
 а) 29;  
 б) 30;  
 в) 32;  
 г) 0.
5. Выберите ответ, соответствующий следующей записи:  $A[8]:=18.$   
 а) значение массива A равно 18;  
 б) значение элемента массива A равно 18;  
 в) значение элемента массива A равно 8;  
 г) в массиве A 18 элементов.
6. В представленном фрагменте программы значения одномерного массива задаются с помощью оператора цикла. Чему будет равно  $A[5]$ ?  
 for i:=1 to 5 do  
 begin  
 $A[2*i-1]:=i;$   
 $A[2*i]:=sqr(i);$   
 end;  
 а) 3;  
 б) 5;  
 в) 7;  
 г) 1.
7. Что можно сказать о массиве, сформированном следующим образом: for i:=1 to 20 do a[i]:=i;?  
 а) массив формируется случайным образом;  
 б) массив заполняется с клавиатуры;  
 в) массив заполняется последовательно увеличивающимися числами;  
 г) массив не заполняется.
8. Укажите способы заполнения массива элементами.  
 а) по определенному закону (формуле);  
 б) из раздела описания переменных (var);  
 в) с помощью функции случайных чисел (random);  
 г) с клавиатуры компьютера (readln).
9. Что определяет для массива  $A[1..n,1..m]$  следующий алгоритм:  
 $S:=0;$   
 for i:= 1 to n do  
 for j:= 1 to m do  
 if  $A[i,j]<0$  then  $S:=S+A[i,j];$   
 а) сумму отрицательных элементов массива;  
 б) минимальный элемент массива;  
 в) количество отрицательных элементов массива;  
 г) сумму минимального и максимального элементов массива.

10. В каком диапазоне будут находиться значения элементов массива А, если он сформирован:  $A[i] := \text{random}(26) + 15$ ?
- а) от 0 до 25;
  - б) от -15 до 40;
  - в) от 15 до 25;
  - г) от 15 до 40.

### Тест 5.3

1. Сформировать массив из 15 элементов с помощью функции случайных чисел в диапазоне от 10 до 80.
- а) `var a:=array [10..80] of integer; a[i]:=random (15);`
  - б) `var a:array [15] of integer; a[i]:=random (71)+10;`
  - в) `var a:=array [10..80] of integer; a[i]:=random (80)+15;`
  - г) `var a:array [1..15] of integer; a[i]:=random (71)+10.`
2. Дана программа:
- ```
program mas;  
var a: array [1..8] of integer;  
c, i: integer;  
begin  
for i:=1 to 8 do readln (a[i]);  
c:=a[1];  
for i:=2 to 8 do  
if c<a[i] then c:=a[i];  
write (c);  
end.
```
- Сколько раз будет выполнен оператор `c:=a[i]` при заданном массиве (2, 7, 6, 8, 3, 19, 1, 10)?
- а) 4;
 - б) 8;
 - в) 7;
 - г) 1.
3. Выберите задачу, которую решает данный фрагмент программы:
- ```
for i:=1 to n do begin
if a[i]>0 then k:=k+1 else s:=s+a[i];
end;
```
- а) определяет сумму и количество положительных элементов массива;
  - б) определяет количество положительных элементов массива;
  - в) определяет количество отрицательных и сумму положительных элементов массива;
  - г) определяет сумму и количество отрицательных элементов массива.
4. Что определяет для массива  $A[1..n, 1..m]$  следующий алгоритм:
- ```
for i:=1 to n do  
for j:=1 to m do  
if i mod 2=0 then A[i,j]=B;
```

- а) четные строки матрицы заменить на В;
 - б) количество четных элементов массива;
 - в) четные столбцы матрицы заменить на В;
 - г) нечетные строки матрицы заменить на В.
5. Данный фрагмент программы формирует элементы массива
- ```
x:=1;
a[1]:=1;
for i:=2 to 5 do
begin
x:=x*2;
a[i]:=a[i-1]+x;
end;
```
- Определите значение пятого элемента массива:  $a[5]=?$
- а) 3;
  - б) 7;
  - в) 15;
  - г) 31.
6. Какой получится вывод после выполнения данной программы, при этом каждый элемент равен своему индексу?
- ```
var A:array [1..10] of integer;
i:byte;
S:integer;
begin
for i:=1 to 10 do read (A[i]);
S:=0;
for i:=1 to 10 do
if i mod 3=0 then S:=S+A[i];
write ('S=', S)
end.
```
- а) 10;
 - б) 9;
 - в) 3;
 - г) 18.
7. Дан массив $C[1..15]$. Определите, какое значение имеет его наибольший элемент после выполнения следующего фрагмента программы:
- ```
for i:=1 to 15 do C[i]:=i+(i mod 5):
```
- а) 15;
  - б) 16;
  - в) 17;
  - г) 18.
8. Значения двух массивов  $A[1..100]$  и  $B[1..100]$  задаются с помощью следующего фрагмента программы:
- ```
for n:=1 to 100 do A[n]:=n-10;
for n:=1 to 100 do B[n]:=A[n]*n;
```

Сколько элементов массива В будут иметь положительные значения?

- а) 90;
- б) 89;
- в) 100;
- г) 50.

9. Добавьте в алгоритм определения среднего арифметического положительных элементов (sr) массива, вместо многоточий, отсутствующую строку:

```
m:=0;
k:=0;
for i := 1 to 10 do
begin
if a[i]>0 then
begin
m:=m+a[i];
k:=k+1
end;
end;
```

.....

- а) $sr:=k/m$;
- б) $sr:=m/k$;
- в) $sr:=k/i$.
- г) $sr:=m/i$.

10. Добавьте в алгоритм определения количества отрицательных элементов (m) массива, вместо многоточий, отсутствующую строку:

```
m:= 0;
for i:=1 to n do begin
if a[i]<0 then .....
```

- ```
end;
a) $m:=m+i$;
```
- б)  $m:=m+1$ ;
  - в)  $m:=m+a[i]$ ;
  - г)  $m:=a[i]$ .

#### Тест 5.4

1. Чему равна сумма  $a[1]$  и  $a[4]$  элементов массива, сформированного следующим образом:

```
for i:=1 to 5 do a[i]:=i*(i+1);
```

- а) 22;
- б) 30;
- в) 40;
- г) 5.

2. Что определяет для массивов  $X[1..n,1..m]$  и  $Y[1..n,1..m]$  следующий алгоритм:



```
for i:=1 to n do
for j:=1 to m do
X[i,j]:=X[i,j]+Y[i,j];?
```

- а) сумма элементов каждой строки матриц X и Y;
- б) сумма элементов главной диагонали матрицы;
- в) сумма матриц X и Y;
- г) количество равных соответствующих элементов матрицы X и матрицы Y.

3. Добавьте в алгоритм определения индекса минимального элемента массива, вместо многоточий, отсутствующие строки:

```
.....
for i := 2 to n do
begin
if a[i] < a[m] then
end;
```

- а) m:=a[1]; m:=i;
- б) m:=1; m:=i;
- в) m:=a[1]; m:=i;
- г) m:=0; m:=m+1.

4. Значения элементов двух массивов размером [1..100] задаются с помощью следующего фрагмента программы:

```
for i:=1 to 100 do A[i]:=50-i;
for i:=1 to 100 do B[i]:=A[i]+49;
```

Сколько элементов массива B будут иметь отрицательные значения?

- а) 100;
- б) 50;
- в) 49;
- г) 1.

5. Значения элементов двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы:

```
for n:=1 to 100 do A[n]:=n-50;
for n:=1 to 100 do B[101-n]:=A[n]*A[n];
```

Какой элемент массива B будет наименьшим?

- а) B[1];
- б) B[50];
- в) B[51];
- г) B[100].

6. При наборе программы вычисления суммы отрицательных элементов массива

```
program mass;
var a: array [1..8] of integer;
s, k: integer;
begin
for k:=1 to 8 do
```

```

readln (a[k]);
s:=0;
for k:=1 to 8 do
if a[k]<0 then s:=s+a[k];
writeln (s);
end.

```

в записи оператора  $s:=s+a[k]$  была допущена ошибка: вместо него был записан оператор  $s:=s+1$ . Каким оказался ответ после исполнения неверной программы, если в качестве элементов массива были введены числа: -1, 3,-2, 4,-5, 6,-7, 8?

- а) -3;
- б) -15;
- в) 4;
- г) 8.

7. Что определяет для массива  $X[1..n,1..m]$  следующий алгоритм

```

v:=0;
for i:=1 to n do
for j:=1 to m do
if X[i, j]=t then
v:=v+1;

```

- а) количество элементов матрицы не равное t;
- б) количество элементов матрицы равные t;
- в) количество равных элементов матрицы;
- г) сумма элементов главной диагонали матрицы.

8. Дан одномерный целочисленный массив А [1..5] из пяти элементов: 3, 6,-4, 7, 1. Определите элементы массива В, который будет получен в результате выполнения алгоритма:

```

i:=1;
while i<=5 do
begin
B[i]:=sqr(A[i])+1;
i:=i+1;
end;

```

- а) 10, 36, 16, 50, 2;
- б) 10, 36, 17,49, 2;
- в) 10, 37, 17, 50, 2;
- г) 9, 37, 17, 49, 1.

9. Какой получится вывод после выполнения данной программы, при этом каждый элемент равен своему индексу?

```

var a:array [1..25] of integer;
i:byte;
p:integer;
begin
for i:=1 to 25 do read (a[i]);
p:=1;

```

```
for i:=1 to 25 do
if (i mod 7>0) and (i<8)
then p:=p*a[i];
write ('p=', p)
end.
```

- а) 1;
- б) 25;
- в) 5760;
- г) 720.

10. Что определяет для массива  $X[1..n, 1..m]$  следующий алгоритм:

```
for i:=1 to n do
for j:=1 to m do
if j mod 2 <> 0 then X[i,j]=1;?
```

(необходимо учитывать, что если не происходит запись переменной, то она автоматически принимает нулевое значение)

- а) формирование матрицы, в которой элементы, стоящие на главной диагонали равны 1, остальные равны 0;
- б) формирование матрицы, в которой элементы, стоящие в нечетных столбцах равны 1, остальные равны 0;
- в) формирование матрицы, в которой элементы, стоящие в четных строках равны 1, остальные равны 0;
- г) формирование матрицы, в которой элементы, стоящие на побочной диагонали равны 1, остальные равны 0.

## **Тема 6: ПРОЦЕДУРЫ И ФУНКЦИИ**

1. Что в Pascal относится к подпрограммам?
  - а) условный оператор и процедура вывода;
  - б) функции и процедуры;
  - в) условный оператор и процедура ввода;
  - г) циклы и массивы.
2. Среди формальных параметров выделяют:
  - а) параметры-значения;
  - б) параметры-выражения;
  - в) параметры-операнды;
  - г) параметры-операции.
3. Подпрограмма-процедура начинается со служебного слова:
  - а) function;
  - б) uses;
  - в) procedure;
  - г) program.
4. Среди формальных параметров выделяют:

- a) function;
  - б) uses;
  - в) procedure;
  - г) program.
5. В каком месте программы должна быть описана подпрограмма?
- a) в основном блоке программы;
  - б) в uses;
  - в) после var;
  - г) в label.
6. Переменные, используемые только в подпрограмме и описанные в разделе описаний переменных подпрограммы, называются:
- a) глобальными;
  - б) локальными;
  - в) параметрическими;
  - г) основными.
7. Между формальными и фактическими параметрами должно существовать соответствие:
- a) только по количеству параметров;
  - б) по порядку их следования и типу данных;
  - в) по количеству параметров и порядку их следования;
  - г) по количеству параметров, порядку их следования и типу данных.
8. Фактические параметры-переменные и соответствующие им формальные параметры процедуры называются:
- a) входными параметрами;
  - б) выходными параметрами;
  - в) центральными параметрами;
  - г) основными параметрами.
9. Фактические параметры-значения и соответствующие им формальные параметры процедуры называются:
- a) выходными параметрами;
  - б) входными параметрами;
  - в) основными параметрами;
  - г) центральными параметрами.
10. Отличительными особенностями подпрограмм-функций от подпрограмм-процедур являются:
- a) вызов процедуры осуществляется внутри выражения по ее имени и наличие у процедуры только одного результата выполнения;
  - б) вызов функции осуществляется внутри выражения по ее имени и наличие у функций только одного результата выполнения;
  - в) описывается сначала процедура, а затем функция;
  - г) вызов процедуры осуществляется внутри выражения по ее имени и наличие у функций только одного результата выполнения.

## Тема 7: СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ И СТРОКИ

### Тест 7.1

1. Функция, копирующая из строки *st* *n* символов в память, начиная с некоторой позиции *p*, является...
  - а) `copy (st,p,n);`
  - б) `val(st,n,m);`
  - в) `delete (st,p,n);`
  - г) такой функции не существует.
2. Функция, возвращающая номер позиции, с которой располагается подстрока *st1* в строке *st2*:
  - а) `copy (st1,p,n);`
  - б) `val(st2,n,m);`
  - в) `delete (st2,p,n);`
  - г) `pos (st1,st2).`
3. Функция, возвращающая длину строки *st*:
  - а) `length (st);`
  - б) `val(st,n,m);`
  - в) `delete (st,p,n);`
  - г) такой функции не существует.
4. Функция, возвращающая цифровой код символа *a*:
  - а) `length (a);`
  - б) `ord(a);`
  - в) `chr(a);`
  - г) `UpCase(a).`
5. Процедура вставки строку *st1* в строку *st2*, в позицию *p*:
  - а) `insert(st1,st2,p);`
  - б) `delete (st,p,n);`
  - в) `pos (st1,st2);`
  - г) `concat (str1,str2).`
6. Процедура удаления *n* знаков части строки *st*, начиная с позиции *p*:
  - а) `insert(st1,st2,p);`
  - б) `val(st,n,p);`
  - в) `delete (st,p,n);`
  - г) `copy (st,p,n).`
7. Процедура перевода целочисленной переменной *n* в строку *st*:
  - а) `val(st,n,m);`
  - б) `str(n,st);`
  - в) `concat (st,n);`
  - г) такой процедуры не существует.
8. Процедура перевода содержимого строки *st* в целочисленную переменную *n*, причем *m* – номер ошибочного символа:
  - а) `delete (st,m,n);`
  - б) `insert(st,m,n);`

- в) `val(st,n,m);`
- г) `copy (st,m,n).`

9. Функция преобразования строчной буквы в прописную:

- а) `Ord(a);`
- б) `Length (a);`
- в) `UpCase(a);`
- г) `Chr(a).`

10. Функция, выполняющая сцепление строк в том порядке, в каком они указаны в списке параметров:

- а) `val(st,n,m);`
- б) `copy (st,m,n);`
- в) `concat (str1,str2,...,strN);`
- г) `delete (st,p,n).`

### Тест 7.2

1. Выберите правильный вариант объявления строковой величины A, состоящей из 25 символов:

- а) `A:string;`
- б) `A: string [25];`
- в) `A:array [1..25] of string.`

2. Выберите правильно описанный формат процедуры `insert`:

- а) строка, подстрока, количество символов;
- б) подстрока, количество символов, символ начала вставки;
- в) подстрока, символ начала вставки, количество символов;
- г) подстрока, строка, символ начала вставки.

3. Каков результат выполнения функции `pos(str1,str2)`:

- а) вставка строки `str1` в строку `str2`, начиная с n-го символа;
- б) вставка строки `str2`, в строку `str1`;
- в) определяет позицию подстроки в строке;
- г) неправильный синтаксис функции.

4. Каков результат выполнения следующих действий?

`S:= '124z';`

`val(S, x, code);`

- а) `x=124;`
- б) `x=0;`
- в) `code=4;`
- г) `code='z'.`

5. Каков результат выполнения следующих действий?

`a:='internet';`

`b:=copy(a,6,3);`

`delete(a,6,3);`

- а) `a='inter'; b='net';`
- б) `a='internet'; b='net';`
- в) `a='inter'; b='ternet';`
- г) `a='net'; b='net'.`

6. В каком случае после выполнения функции pos( ) будет результат 0?
- а) когда найден символ, стоящий на нулевом месте;
  - б) когда не найден символ;
  - в) когда найден символ, стоящий на последнем месте;
  - г) когда найден символ, стоящий на первом месте.
7. Задана строка s:=«география». Как можно получить из нее строку «граф-фити»?
- а) delete(s,1,3); a:=copy(s,1,5); c:=copy(a,5,1); b:=concat(a, 'т',c );
  - б) delete(s,1,3); a:=copy(s,1,5); insert ('ф',a,4); b:=concat(a, 'ти');
  - в) delete(s,1,3); a:=copy(s,1,5); b:=concat(a, 'т', a[5]);
  - г) b:=copy(a,4,5)+'ти'.
8. Каков результат после выполнения программы?
- ```
var a,b,c,d: char;
begin
read (a,b,c,d);
write(a,':',b,',' ,c, ' ',d, '?')
end.
```
- а) a,b,c,d
 - б) a:b/c d?
 - в) a:,b/,c ,d?
 - г) a: b/ cd.
9. Какой результат будет выведен на экран?
- ```
str1:= 'abcdef';
str2:= 'ABCDEF';
insert(str1,str2,5);
insert(str2,str1,4);
k1:=length(str2);
writeln (str2);
writeln (str1);
writeln (k1);
```
- а) ABCDabcdefEF; abcABCDabcdefEFdef; 12;
  - б) abcABCDabcdefEFdef; ABCDabcdefEF; 18;
  - в) ABCabcdefEF; ABCDabcdefEFdefabc; 12;
  - г) ABCDabcdefEF; abcABCDabcdefEFdef; 18.
10. Какой результат будет выведен на экран?
- ```
str1:= 'АБВГДЕЖЗИЙКЛ';
str2:= 'ABCDEFGH';
str3:=copy(str1,5,3);
writeln(str3);
writeln(copy(str2,1,4));
```
- а) ДЕЖ; ABCD;
 - б) ABCD; ДЕЖ;
 - в) ДЕ; CD;
 - г) ЕЖ; ВС.

Тема 8: ОРГАНИЗАЦИЯ БИБЛИОТЕК. СТАНДАРТНЫЕ БИБЛИОТЕЧНЫЕ МОДУЛИ И МОДУЛИ ПОЛЬЗОВАТЕЛЯ. СТРУКТУРА UNIT-A

1. Для организации отдельных тематических библиотек в Pascal введены:
 - а) подпрограммы-процедуры;
 - б) подпрограммы-функции;
 - в) модули;
 - г) справочники.
2. Заголовок модуля состоит из зарезервированного слова:
 - а) uses;
 - б) interface;
 - в) unit;
 - г) implementation.
3. Связь модуля с основной программой устанавливается специальным предложением:
 - а) uses;
 - б) interface;
 - в) unit;
 - г) implementation.
4. Как охарактеризовать модуль?
 - д) часть основной программы;
 - е) автономно компилируемая программная единица;
 - ж) подпрограмма-функция;
 - з) подпрограмма-процедура.
5. Предложение uses <список модулей> должно стоять сразу после:
 - а) операторных скобок;
 - б) заголовка программы;
 - в) раздела описаний;
 - г) блока основной программы.
6. Интерфейсная часть открывается зарезервированным словом:
 - а) unit;
 - б) implementation;
 - в) interface;
 - г) begin.
7. Исполняемая часть модуля начинается зарезервированным словом:
 - а) interface;
 - б) unit;
 - в) implementation;
 - г) begin.
8. Подключить модуль к основной программе можно с помощью:
 - а) unit;
 - б) uses;

- в) program;
- г) procedure.

9. Список модулей, с которыми устанавливается связь, перечисляются:

- а) через двоеточие;
- б) через запятую;
- в) в блоке основной программы;
- г) через точку с запятой.

10. Впервые специализированная синтаксическая конструкция модуля была предложена:

- а) Д. Парнасом;
- б) Н. Виртом;
- в) Джоном фон Нейманом;
- г) П. Дуровым.

Глава IV. ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

Зачетные задания

Задача 1: составить программу, которая формирует массив из 12 случайных целых чисел, принадлежащих промежутку $[-40;40]$, и вычисляет сумму и произведение положительных чисел. Вычисление суммы и произведения положительных чисел реализовать с помощью подпрограммы.

Задача 2: составить программу, которая формирует массив из 7 случайных целых чисел, принадлежащих промежутку $[1;5]$, и вычисляет произведение нечетных чисел. Вычисление произведения нечетных чисел реализовать с помощью подпрограммы.

Задача 3: составить программу, которая формирует массив из 10 случайных целых чисел, принадлежащих промежутку $[0;8]$, и вычисляет частное четных чисел. Вычисление частного четных чисел реализовать с помощью подпрограммы.

Задача 4: составить программу, которая формирует массив из 15 случайных целых чисел, принадлежащих промежутку $[-4;8]$, и вычисляет среднее арифметическое чисел, кратных 3. Вычисление среднего арифметического чисел, кратных 3, реализовать с помощью подпрограммы.

Задача 5: составить программу, которая формирует массив из 40 случайных целых чисел, принадлежащих от 0 до 20 и вычисляет среднее арифметическое нечетных чисел. Вычисление среднего арифметического нечетных чисел реализовать с помощью подпрограммы.

Задача 6: составить программу, которая формирует массив из 12 случайных целых чисел, принадлежащих промежутку $[-20;20]$, заменяет отрицательные числа на -1 и выводит значения элементов массива до и после замены (реализовать с помощью подключения соответствующего модуля).

Задача 7: составить программу, которая формирует массив из 7 случайных целых чисел, принадлежащих промежутку $[-6;4]$, заменяет все четные значения элементов массива квадратами их номеров и выводит значения элементов массива до и после замены (реализовать с помощью подключения соответствующего модуля).

Задача 8: составить программу, которая формирует массив из 10 случайных целых чисел, принадлежащих промежутку $[2;10]$, заменяет все отрицательные числа их квадратами, а неотрицательные – значениями квадратного корня и выводит значения элементов массива до и после замены (реализовать с помощью подключения соответствующего модуля).

Задача 9: составить программу, которая формирует массив из 15 случайных целых чисел, принадлежащих промежутку $[-20;10]$, и четные числа делит на два, а нечетные – умножает на три и выводит значения элементов массива до и после замены (реализовать с помощью подключения соответствующего модуля).

Задача 10: составить программу, которая формирует массив из 20 случайных целых чисел, принадлежащих промежутку $[-20;20]$, и отрицательные

числа возводит в квадрат, а из неотрицательных извлекает квадратный корень и выводит значения элементов массива до и после замены (реализовать с помощью подключения соответствующего модуля).

Задача 11: составить программу, которая формирует массив из 13 случайных целых чисел, принадлежащих промежутку $[2;18]$, и увеличивает в 2 раза числа, кратные 5, а остальные уменьшает в 2 раза и выводит значения элементов массива до и после замены (реализовать с помощью подпрограммы).

Задача 12: составить программу, которая формирует массив из 15 случайных целых чисел, принадлежащих промежутку $[0;50]$, и выводит на экран нечетные элементы массива и их индексы (реализовать с помощью подпрограммы).

Задача 13: составить программу, которая формирует массив из 16 случайных целых чисел от -30 до 20 и находит индексы четных элементов (реализовать с помощью подпрограммы).

Задача 14: составить программу, которая формирует массив из 18 случайных целых чисел от -25 до 25 и подсчитывает количество отрицательных элементов (реализовать с помощью подпрограммы).

Задача 15: составить программу, которая формирует массив из 11 случайных целых чисел от -18 до 5 и подсчитывает количество элементов, кратных 3 (реализовать с помощью подпрограммы).

Перечень литературы по дисциплине «Основы алгоритмизации и программирования»

Основная литература

1. Алексеев, Е.Р. Турбо Паскаль 7.0. / Е.Р. Алексеев, О.В. Чеснокова. – Москва : НТ Пресс, 2006. – 314 с.
2. Батан, С.Н. Сборник задач по программированию : учебно-методические материалы / С.Н. Батан, Н.В. Кожуренко. – Могилев : МГУ имени А.А. Кулешова, 2015. – 83 с.
3. Задачи по программированию / под ред. С.М. Окулова. – Москва : БИНОМ. Лаборатория знаний, 2006. – 820 с.
4. Лубашева, Т.В. Основы алгоритмизации и программирования : учеб. пособие / Т.В. Лубашева, Б.А. Железко. – Минск : РИПО, 2016. – 378 с.
5. Мельников, О.И. Методы алгоритмизации : учебное пособие для 10–11 классов общеобразовательной школы с углубленным изучением информатики / О.И. Мельников, В.М. Котов. – Минск : Нар. асвета, 2000. – 221 с.
6. Окулов, С. М. Основы программирования / С.М. Окулов. – 3-е изд. – Москва : БИНОМ. Лаборатория знаний, 2006. – 440 с.
7. Окулов, С. М. Программирование в алгоритмах / С.М. Окулов. – 3-е изд. – Москва : БИНОМ, 2007. – 383 с.
8. Окулов, С.М. Программирование в алгоритмах / С.М. Окулов. – Москва : Лаборатория знаний, 2002. – 341 с.

Дополнительная литература

1. Delphi 7 / В. Гофман, Е. Мещеряков, В. Никифоров ; под общ. ред. А.Д. Хомоненко. – Санкт-Петербург : ВHV-Санкт-Петербург, 2004. – 1216 с.
2. Delphi 7. Учебный курс / С.И. Бобровский . – Санкт-Петербург : Питер, 2004. – 736 с.
3. Базы данных : учебник для вузов / под ред. А.Д. Хомоненко. – Санкт-Петербург : КОРОНА Принт, 2000. – 416 с.
4. Батан, Л.В. DELPHI 7 : лабораторный практикум / Л.В. Батан. – Могилев : МГУ имени А.А. Кулешова, 2009. – 80 с.
5. Губина, Т.Н. Язык программирования Паскаль : лабораторный практикум. Часть 1 : учебное пособие / Т.Н. Губина, В.А. Дякина, М.А. Губин. – Елец : ЕГУ им. И.А. Бунина, 2012. – 102 с.
6. Давыдов, В.Г. Программирование и основы алгоритмизации : учебное пособие / В.Г. Давыдов. – 2-е изд., стереотип. – Москва : Высш. шк., 2005. – 447 с.
7. Демидов, Д.В. Основы программирования в примерах на языке Паскаль : учебное пособие / Д.В. Демидов. – Москва : НИЯУ МИФИ, 2010. – 172 с.
8. Живицкая, Е. Н. Информационные технологии : учебное пособие / Е.Н. Живицкая, И.Г. Орешко, Э.С. Иванова. – Минск : Беларусь, 2008. – 205 с.
9. Златопольский, Д.М. Сборник задач по программированию / Д.М. Златопольский . – 2-е изд. – Санкт-Петербург : ВHV-Санкт-Петербург, 2007. – 240 с.
10. Зуев, Е.А. Система программирования Turbo Pascal / Е.А. Зуев. – Москва : Диалог : Радио и связь, 1992. – 288 с.

11. Зуев, Е.А. Язык программирования Turbo Pascal 6.0 / Е.А. Зуев. – Москва : Унитех, 1992. – 298 с.
12. Информатика : 11 класс : учебное пособие с углубленным изучением информатики для общеобразовательных школ с русским языком обучения / А.И. Павловский [и др.]. – Минск : Нар. асвета, 2001. – 302 с.
13. Кадырова, Г.Р. Основы алгоритмизации и программирования : учебное пособие / Г.Р. Кадырова. – Ульяновск : УлГТУ, 2014. – 95 с.
14. Колдаев, В.Д. Численные методы и программирование : учебное пособие для студентов учреждений среднего профессионального образования / В.Д. Колдаев. – Москва : Форум: ИНФРА-М, 2009. – 336 с.
15. Кудрявцев, А.С. Программирование в Delphi : учебное пособие / А.С. Кудрявцев. – Санкт-Петербург : ГОУВПО СПбГТУРП, 2011. – 102 с.
16. Культин, Н.Б. Основы программирования в Delphi 7 / Н.Б. Культин. – Санкт-Петербург : ВHV-Санкт-Петербург, 2004. – 608 с.
17. Культин, Н.Б. Основы программирования в Delphi XE / Н.Б. Культин. – Санкт-Петербург : БХВ-Петербург, 2011. – 413 с.
18. Культин, Н.Б. Основы программирования в Turbo Delphi / Н.Б. Культин. – Санкт-Петербург : ВHV-Санкт-Петербург, 2007. – 384 с.
19. Лазицкас, Е.А. Базы данных и системы управления базами данных : учеб. пособие / Е.А. Лазицкас, И.Н. Загумённикова, П.Г. Гилевский. – Минск : РИПО, 2016. – 268 с.
20. Ляхович, В.Ф. Основы информатики : учебное пособие для студентов средних специальных учебных заведений / В.Ф. Ляхович. – Ростов-на-Дону : Феникс, 2000. – 608 с.
21. Павловский, А.И. Дидактические материалы по информатике: 10 класс / А.И. Павловский, А.Е. Пупцев. – Минск : Нар. асвета, 2004. – 48 с.
22. Пильщиков, В.Н. Сборник упражнений по языку Паскаль : учебное пособие для вузов / В.Н. Пильщиков. – Москва : Наука, 1989. – 160 с.
23. Turbo Паскаль 7.0 / Е.Р. Алексеев [и др.]. – 2-е изд. – Москва : НТ Пресс, 2006. – 272 с.
24. Тюкачев, Н.А. Программирование в Delphi для начинающих : учебное пособие для студентов вузов / Н.А. Тюкачев, К.С. Рыбак, Е.Е. Михайлова. – Санкт-Петербург : ВHV-Санкт-Петербург, 2007. – 672 с.
25. Фаронов, В.В. Delphi 2005 : разработка приложений для баз данных и Интернета / В.В. Фаронов. – Санкт-Петербург : ПИТЕР, 2006. – 603 с.
26. Фаронов, В.В. Система программирования Delphi / В.В. Фаронов. – СПб. : БХВ-Петербург, 2003. – 912 с.
27. Фаронов, В.В. Turbo Паскаль / В.В. Фаронов. – М. : МВТУ-ФЕСТО ДИДАКТИК, 1992. – 304 с.

ЗАКЛЮЧЕНИЕ

При преподавании дисциплины «Основы алгоритмизации и программирование» цикловой комиссией информационных технологий в качестве базового принят язык программирования Pascal. Язык Pascal выбран в качестве базового алгоритмического языка ввиду относительно небольшого количества его конструкций, структурированности, разделения секций описания и реализации, широких возможностей написания самых различных по направленности программ.

Язык Pascal обладает многими достоинствами в плане обучения, среди которых можно выделить следующие: простота и ясность конструкций; высокая типизация данных; контроль типов; возможность построения новых типов данных; возможность достаточно полного контроля правильности программы на этапе компиляции и во время ее выполнения; возможность удовлетворения требованиям структурного программирования; гибкость и надежность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Губина, Т.Н. Язык программирования Паскаль : лабораторный практикум : учебное пособие / Т. Н. Губина, В. А. Дякина, М. А. Губин. – Елец : ЕГУ им. И.А. Бунина, 2012. – Часть 1. – 102 с.
2. Демидов, Д.В. Основы программирования в примерах на языке Паскаль : учебное пособие / Д.В. Демидов. – Москва : НИЯУ МИФИ, 2010. – 172 с.
3. Задачи по программированию / под ред. С.М. Окулова. – Москва : БИНОМ. Лаборатория знаний, 2006. – 820 с.
4. Кадырова, Г.Р. Основы алгоритмизации и программирования : учебное пособие / Г.Р. Кадырова. – Ульяновск : УлГТУ, 2014. – 95 с.
5. Лабораторный практикум по программированию на языке Паскаль : учебное пособие / под общ. ред. Л.В.Найхановой и Н.Ц. Бильгаевой. – 3-е изд. перераб. и доп. – Улан-Удэ, 2004. – 176 с.
6. Лубашева, Т.В. Основы алгоритмизации и программирования : учеб. пособие / Т.В. Лубашева, Б.А. Железко. – Минск : РИПО, 2016. – 378 с.
7. Окулов, С.М. Основы программирования / С.М. Окулов. – 3-е изд. – Москва : БИНОМ. Лаборатория знаний, 2006. – 440 с.

Учебное издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

РАЗДЕЛ PASCAL

Учебно-методический комплекс

Составитель

Новашинская Светлана Сергеевна

Технический редактор *А.Л. Позняков*
Компьютерная верстка *А.Л. Позняков*
Корректор *И.Г. Латушкина*

Подписано в печать .03.2019.
Формат 60x84/16. Гарнитура Times New Roman Cyr.
Усл.-печ. л. 6,5. Уч.-изд. л. 4,0. Тираж 60 экз. Заказ № .

Учреждение образования “Могилевский государственный университет
имени А.А. Кулешова”, 212022, Могилев, Космонавтов, 1
Свидетельство ГРИИРПИ № 1/131 от 03.01.2014 г.

Отпечатано в отделе оперативной полиграфии
МГУ имени А. А. Кулешова. 212022, Могилев, Космонавтов