Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра конструирования и производства радиоаппаратуры

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 3-4 по дисциплине
«Информатика» для студентов направления
12.03.01 «Приборостроение» (профиль «Приборостроение»)
всех форм обучения

Воронеж 2021

УДК 004.432.2
ББК 32.97

Составители:
д-р. техн. наук М.А. Ромащенко,
канд. техн. наук А.А. Пирогов,
И.В. Свиридова

Методические указания к выполнению лабораторных работ № 3-4 по дисциплине «Информатика» для студентов направления 12.03.01 «Приборостроение» (профиль «Приборостроение») всех форм обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост. М.А. Ромащенко, А.А. Пирогов, И.В. Свиридова. Воронеж: Изд-во ВГТУ, 2021, 36 с.

Методические указания предназначены для выполнения лабораторных работ № 3-4 по дисциплине «Информатика» студентами очной и заочной форм обучения.

Предназначены для студентов первого курса обучения.

Методические указания подготовлены в электронном виде и содержатся в файле ЛР3-4.pdf.

Ил. 11. Библиогр.: 4 назв.

УДК 004.432.2
ББК 32.97

Рецензент - О. Ю. Макаров, д-р техн. наук, проф. кафедры конструирования и производства радиоаппаратуры ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

1. ЛАБОРАТОРНАЯ РАБОТА №3

АЛГОРИТМЫ. ОСНОВНЫЕ ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ ПАСКАЛЬ.

Цель работы: изучить виды алгоритмов, их свойства и классификацию, получить навыки практической реализации алгоритмов. Изучить основные возможности языка программирования Паскаль. Ознакомиться со структурой, синтаксисом и семантикой программы на языке Паскаль.

Время работы: 8 часов.

1.1. Домашние задания и методические указания по их выполнению

Задание 1 – получить представление об алгоритмах, их свойствах и классификации.

Алгоритм – точное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

Основные свойства алгоритмов:

1. ***Понятность для исполнителя*** - чтобы алгоритм можно было выполнить, он должен быть понятен исполнителю, т.о. при формулировке алгоритма необходимо учитывать возможности и особенности исполнителя, на которого рассчитан алгоритм.
2. ***Конечность (дискретность)*** – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).
3. ***Определенность*** – каждый шаг алгоритма должен быть четко и недвусмысленно определен и не допускать произвольной трактовки исполнителем.
4. ***Результативность*** – цель выполнения алгоритма это получение

результата (результатов), имеющего вполне определенное отношение к исходным данным.

5. ***Массовость*** – возможность применять один и тот же алгоритм для решения целого класса однотипных задач, различающихся исходными данными.

6. ***Эффективность*** – каждый шаг алгоритма должен быть выполнен точно и за конечное разумное время.

По ***форме представления*** алгоритмы классифицируются на

- словесные (записи на естественном языке);
- графические (изображения из графических символов);
- псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке);
- программные (тексты на языках программирования).

При графическом представление алгоритм изображается в виде последовательности связанных между собой функциональных блоков. Блочные символы соединяются линиями переходов, определяющими очередность выполнения действий. На рис.1 приведены наиболее часто употребляемые символы.

Название символа	Обозначение и пример заполнения	Пояснение
Процесс	x=(a-b)/sin(l)	Вычислительное действие или последовательность действий
Решение	да a<b нет	Проверка условий
Модификация	i=1,50,2	Начало цикла
Предопределенный процесс	Расчет параметров	Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод	Ввод a,b,c	Ввод-вывод в общем виде
Пуск-останов	Начало	Начало, конец алгоритма, вход и выход в подпрограмму
Документ	Печать a,c	Вывод результатов на печать

Рис. 1. Виды функциональных блоков при графическом представлении алгоритма

Соотношение сторон функциональных блоков должно быть как b=2*a, где a=10, 15, 20 мм.

В информатике универсальным исполнителем алгоритмов является компьютер. ***Исполнитель алгоритма*** — это устройство управления, соединенное с набором инструментов. Устройство управления понимает алгоритмы и организует их выполнение, командуя соответствующими инструментами.

Задание 2 – ознакомиться с понятием «язык программирования» и классификацией языков программирования.

Язык программирования это формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя. ***Синтаксис языка программирования*** – это набор правил построения конструкций языка. ***Семантика языка программирования*** – это совокупность значений (смысл) всех конструкций языка.

Языки программирования подразделяются на

машинно-ориентированные (языки низкого уровня),

машинно-независимые языки (языки высокого уровня).

В ***машинно–ориентированных языках*** типы и структуры данных, операции над данными и порядок выполнения операторов определяются организацией и функционированием ЭВМ. Машинно-ориентированные языки характеризует:

- высокое качество создаваемых программ (компактность и скорость выполнения);
- возможность использования конкретных аппаратных ресурсов;
- предсказуемость объектного кода и заказов памяти;
- для составления эффективных программ необходимо знать систему команд и особенности функционирования данной ЭВМ;
- трудоемкость процесса составления программ плохо защищенного от появления ошибок;
- низкая скорость программирования;

- невозможность непосредственного использования программ, составленных на этих языках, на ЭВМ других типов.

Машинно-ориентированные языки по степени автоматического программирования подразделяются на классы:

- машинный язык;
- языки символического кодирования;
- автокоды;
- макрос.

Машинно–независимые языки базируются на концепции абстрактных типов данных и абстрактных операциях, что обеспечивает независимость языка от организации и функционирования ЭВМ и дает возможность разработчикам программного обеспечения объявлять новые типы данных в разрабатываемых программах. Т.о., командные последовательности (процедуры, подпрограммы), часто используемые в машинных программах, представлены в высокоуровневых языках отдельными операторами. Программист получает возможность не расписывать в деталях вычислительный процесс на уровне машинных команд, а сосредоточиться на основных особенностях алгоритма. Машинно–независимые языки подразделаюися на:

- проблемно–ориентированные языки;
- универсальные языки;
- диалоговые языки;
- непроцедурные языки.

Задание 3 – закрепить знания о понятиях транслятор, компилятор, интерпретатор.

Транслятор (англ. translator — переводчик) – это программа-переводчик, она преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд. Трансляторы реализуются в виде компиляторов или интерпретаторов, которые с точки зрения выполнения работы существенно различаются.

Компилятор (англ. compiler — составитель, собиратель)

читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Интерпретатор (англ. interpreter — истолкователь, устный переводчик) переводит и выполняет программу строка за строкой.

После того, как программа откомпилирована, ни сама исходная программа, ни компилятор более не нужны. В то же время программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом очередном запуске программы. Откомпилированные программы работают быстрее, но интерпретируемые проще исправлять и изменять. Иногда для одного языка имеется и компилятор, и интерпретатор. В этом случае для разработки и тестирования программы можно воспользоваться интерпретатором, а затем откомпилировать отлаженную программу, чтобы повысить скорость ее выполнения.

Системы программирования – это комплексы программ и прочих средств, предназначенных для разработки программ и их эксплуатации на конкретном языке программирования для конкретного вида ЭВМ.

Обычно система программирования включает:

- текстовый редактор,
- отладчик,
- транслятор,
- компоновщик (редактор связей),
- программа обеспечивающая запуск программы.

Задание 4 – ознакомиться с процессом разработки программы в системе программирования Турбо Паскаль.

Алгоритмический язык высокого уровня Паскаль был разработан в конце 60-х годов профессором Н.Виртом. Он был создан специально для обучения программированию.

Система программирования Турбо Паскаль разработана американской корпорацией Борланд в конце 80-х начале 90-х и предназначена для создания программ, работающих под

управлением ОС MS-DOS. В процессе разработки программы обычно создаются и используются следующие файлы:

- файл с расширением .pas, содержащий исходный текст программы;
- файл с расширением .tpu, в который помещается результат компиляции программы;
- файл с расширением .tpl, содержащий стандартные подпрограммы;
- файл с расширением .exe, содержащий готовую к работе программу.

На рис. 2 показана схема процесса разработки программ в Турбо Паскале.

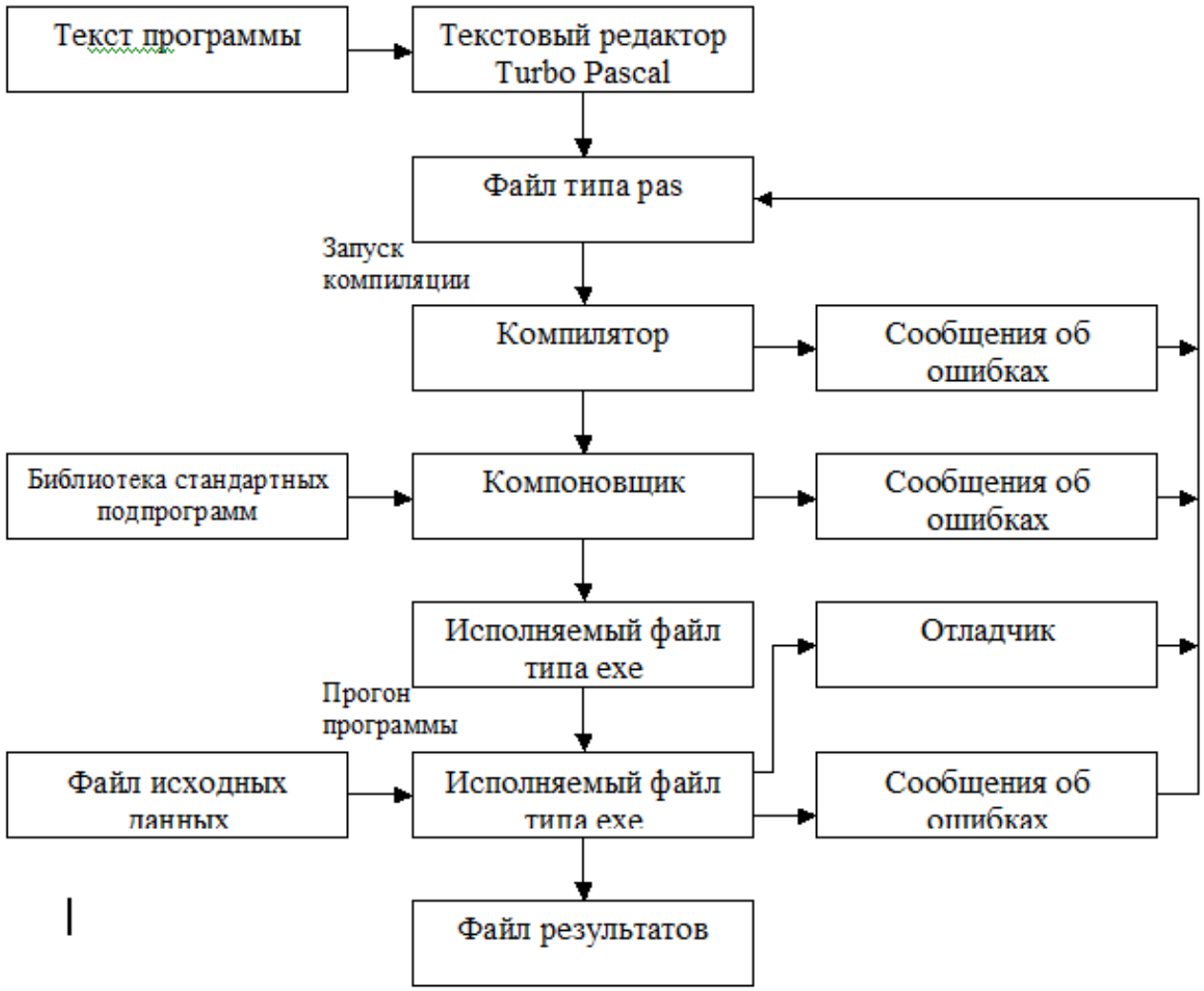

Рис. 2. Схема процесса разработки программ в Турбо Паскале

Перечислим основные «горячие» клавиши системы программирования Турбо Паскаль:

F1 – вызвать справку;

F2 – записать файл из активного окна редактора на диск;

F3 – прочитать файл с диска и поместить его в окно редактора;

F5 – развернуть текущее окно редактора на весь экран или вернуть ему прежние размеры;

F6 – сделать активным следующее окно редактора;

F9 – откомпилировать программу;

F10 – сделать активным главное меню;

Ctrl+F9 – откомпилировать и скомпоновать программу, выполнить ее прогон;

Alt+X – завершить работу с Турбо Паскалем.

При компиляции Турбо Паскаль автоматически обнаруживает все синтаксические ошибки и их исправление, как правило, не вызывает проблем. Семантические (смысловые) ошибки компилятор обнаружить не может, для их поиска предназначен отладчик. Суть его работы заключается в пошаговом исполнении программы, с возможностью контроля значений переменных или выражений. Для работы с отладчиком создаются т.н. точки останова и окна наблюдения. В точках останова нормальное выполнение программы прерывается и управление получает отладчик, при этом в окнах наблюдения отображаются текущие значения переменных или выражений.

Для управления отладчиком используются следующие «горячие» клавиши:

F4 – начать прогон программы и остановиться перед выполнением строки исходного кода, в которой располагается курсор;

F7 – начать или продолжить прогон программы, а если в текущей строке программы есть обращение к подпрограмме, остановиться перед выполнением первого ее оператора;

F8 – начать или продолжить прогон программы, а если в текущей строке программы есть обращение к подпрограмме, не

прослеживать ее работу;

Ctrl+F2 – завершить работу отладчика.

Задание 5 – ознакомиться со структурой программой на языке Паскаль.

Структура программы на языке Паскаль имеет следующий вид:

```
Program Test;
{Раздел описаний}
begin
{Раздел исполняемых операторов}
end.
```

Зарезервированное слово Program означает, что данная программная единица является программой (еще бывают модули, процедуры, функции). Test это название программы и может быть любым.

В разделе описаний должны содержаться описания всех идентификаторов, используемых в разделе исполняемых операторов, объявляются идентификаторы типов, объектов, констант, переменных, а также метки, процедуры и функции.

Описание типов и объектов должно начинаться зарезервированным словом type, описание констант – const, переменных – var и меток label, например:

```
type
    DigType=set of `0`..`9`;
    StrType=String [40];
const
    N=100;
    EPS=1e-9;
var
    x, y: real;
    st: StrType;
```

label
lb1, lb2;

Задание 6 – ознакомиться с алфавитом и идентификаторами языка Паскаль.

Алфавит языка Паскаль включает буквы, цифры, в т.ч. шестнадцатеричные, специальные символы, пробелы, зарезервированные слова и специальные директивы.

Буквы – это буквы латинского алфавита от a до z и от A до Z, а также знак подчеркивания _. В Паскале нет различия между заглавными и строчными буквами алфавита, если только они не входят в строковые и символьные выражения.

Цифры – арабские цифры от 0 до 9.

Специальные символы – знаки + - * / = и т.д.

Зарезервированные слова – используются для описания операторов, данных и других языковых конструкций. Они придают тексту программы более «читабельный» вид, приближая его к тексту, написанному на ествественном английском языке. Зарезервированные слова не могут использоваться в качестве идентификаторов. Например: *program, procudere, for, if, array* и т.д.

Идентификаторы – это имена констант, переменных, меток, типов, объектов, процедур и функций. Они могут иметь произвольную длину, но значащими (уникальными в области определения) являются только первые 63 символа. Идентификатор всегда начинается буквой или подчеркиванием, за которой могут следовать и буквы и цифры. Пробелы и спец.символы не могут входить в идентификатор.

Примеры правильных идентификаторов:
a
ALPHA
MyProgramIsBestProgram
Date_27_sep_39
_beta

Пример неправильных идентификаторов

1Program {начинается цифрой}
block#1 {содержит спец.символ}
My Prog {содержит пробел}
mod {зарезервированное слово}

Задание 7 – ознакомиться с константами, выражениями и операциями языка Паскаль.

В качестве констант могут использоваться целые, вещественные и шестнадцатеричные числа, логические константы, символы, строки символов, конструкторы множеств и признак неопределенного указателя *NIL*.

Целые числа записываются со знаком или без него по обычным правилам и могут иметь значение от –2 147 483 648 до +2 147 483 647. В случае необходимости записать большее число необходимо использовать вещественные числа.

Вещественные числа записываются со знаком или без него с использованием десятичной точки или экспоненциальной части. Экспоненциальная часть начинается символом ***e*** или ***E***, за которым следует знак + или – и десятичный порядок. Символ ***e*** читается как «умножить на 10 в степени». Если в записи вещественного числа присутствует десятичная точка, то перед ней должна стоять хотя бы одна цифра. Если используется символ ***e*** за ним должна следовать хотя бы одна цифра.

Шестнадцатеричное число состоит из шестнадцатеричных цифр, которым предшествует знак $. Диапазон от $00000000 до $FFFFFFFF.

Логическая константа – это либо слово *false* (ложь), либо слово *true* (истина).

Символьная константа – любой символ заключенный в апострофы `z`, `a`

Строковая константа – любая последовательность символов заключенная в апострофы. *`Это строка символов`*.

Выражения языка Паскаль. Основными элементами из которых конструируется исполняемая часть программы, являются константы, переменные и обращения к функциям. Каждый из этих элементов характеризуется своим значением и принадлежит к какому-либо типу данных. С помощью знаков операций и скобок из них можно составлять ***выражения***, которые фактически представляют собой правила получения новых значений. Значение такого выражения имеет тот же тип, что и элемент. В более общем случае выражение состоит из нескольких элементов (операндов) и знаков операций, а тип его значение определяется типов операндов и видом примененных к ним операций.

Примеры
*(a+b)*c*
sin(t)
a>2
not Flag and (a=b)

Операции языка Паскаль

В Паскале определены следующие операции
- унарные операции: *not @*;
- мультипликативные операции: ** / div mod and shl shr*;
- аддитивные операции: *+ - or xor*;
- операции отношения: *= <> < > <= >= in*.

Приоритет операций убывает в указанном порядке, т.е. высшим приоритетом обладают унарные операции, низшим – операции отношения. Операции равного приоритета выполняются из условия оптимизации кода.

В Паскале определены следующие логические операции
- *not* – логическое НЕ;
- *and* – логическое И;
- *or* – логическое ИЛИ;
- *xor* – исключающее ИЛИ.

Задание 8 – ознакомиться с типами данных языка Паскаль.

Любые данные, т.е. константы, переменные, значения функций или выражения, в Паскале характеризуются своими типами. ***Тип*** определяет множество допустимых значений, которые может иметь тот или иной объект, а также множество допустимых операций, которые применимы к нему. Кроме того тип определяет формат внутреннего представления данных в памяти компьютера. ***Мощностью типа*** называется суммарное количество всех возможных его значений.

Порядковые типы отличаются тем, что каждый из них имеет конечное количество возможных значений. Это значение можно определенным образом упорядочить и, следовательно, с каждым из них можно сопоставить некоторое целое число – порядковый номер.

Вещественные типы, тоже имеют конечное количество, которое определяется форматом внутреннего представления вещественного числа, однако это количество настолько велико, что сопоставить с каждым из них целое число (его номер) не представляется возможным.

Все возможные типы данных языка Паскаль и их связь представлена на рис. 3.

Диапазон возможных значений ***целых типов*** зависит от их внутреннего представления, которое может занимать один, два или четыре байта.

Таблица 1

Диапазон возможных значений целых типов

Название	Длина, байтов	Диапазон значений	Мощность типа
Byte	1	От 0 до 255	256
ShortInt	1	От –128 до +127	256
Word	2	От 0 до 65535	65536
Integer	2	От –32768 до +32767	65536
LongInt	4	От –2 147 483 648 до +2 147 483 647	4294497296

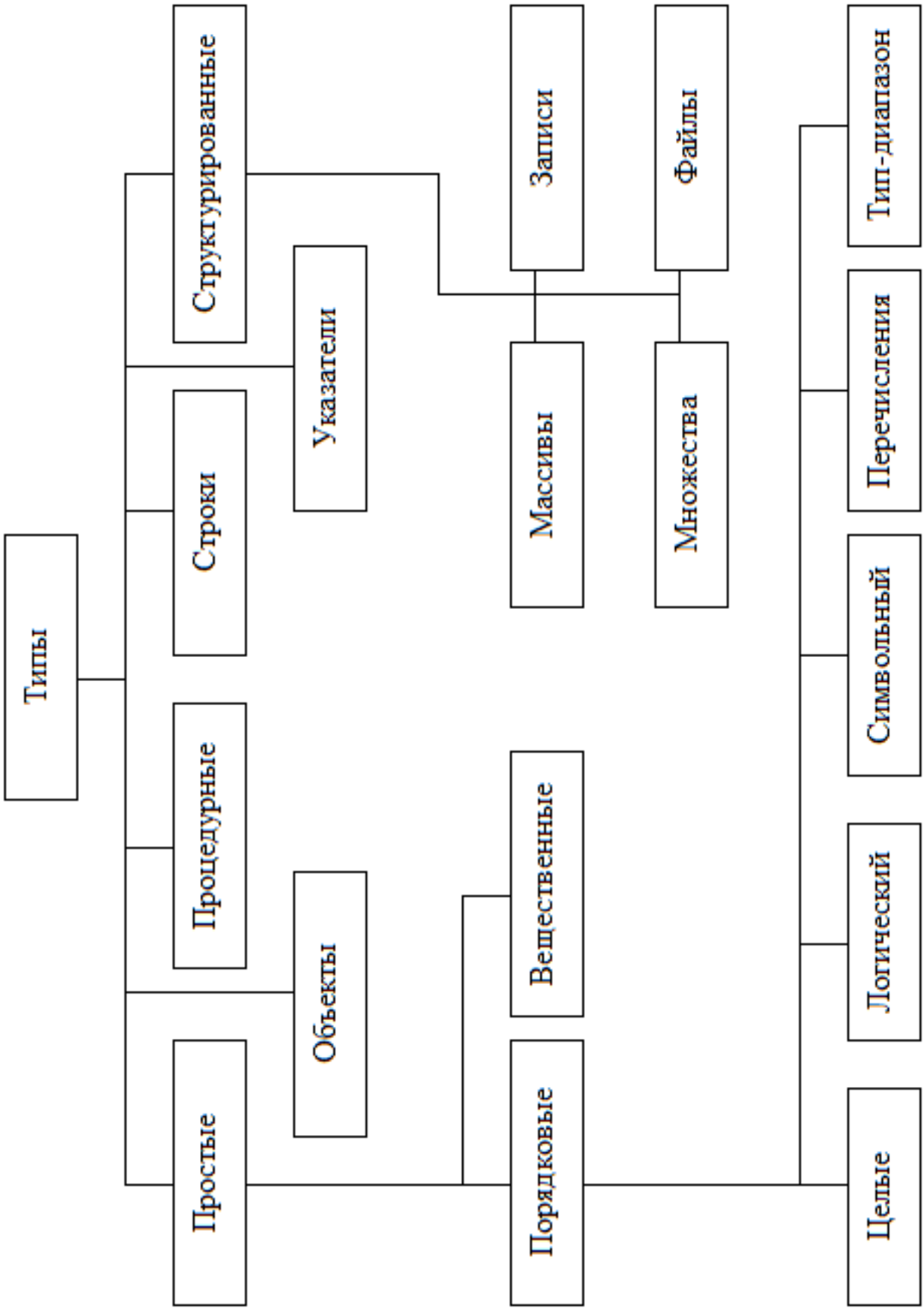

Рис. 3. Типы данных языка Паскаль

Значениями ***логического типа*** может быть одна из предварительно объявленных констант False (ложь) или True (истина). Описывается как Boolean.

Значением ***символьного типа*** является множество всех символов компьютера. Каждому символу приписывается целое число в диапазоне от 0 до 254. Для кодировки используется код ASCII, каждому символу соответствует один байт. Описывается как Char.

Перечисление или ***перечисленный тип***, задается перечислением тех значений, которые он может получать. Каждое значение именуется некоторым идентификатором и располагается в списке, пример:

```
type
type_of_colors=(red, white, blue);
var
    color: type_of_colors;
```

Тип-диапазон есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона. Тип-диапазон задается границами своих значений внутри базового типа - *<мин.знач>..<макс.знач>*. Пример:

```
var
    date: 1..31;
    month: 1..12;
```

Значения вещественных типов определяют произвольное число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

Таблица 2

Диапазон возможных значений вещественных типов

Название	Длина, байтов	Кол-во значащих цифр	Диапазон десятичного порядка
Single	4	7-8	-45..+38
Real	6	11-12	-39..+38

Продолжение табл. 2

Double	8	14-16	-324..+308
Extended	10	19-20	-4951..+4932
Comp	8	19-20	-2e63+1..+2e63-1

Массив это упорядоченная последовательность компонентов одного типа, объединенных под одним именем. Эти компоненты можно легко упорядочить и обеспечить доступ к любому из них простым указанием его порядкового номера. Описание производится так *<имя>: array [<сп.инд.типов>] of <тип>*. Здесь *<имя>* - правильный идентификатор; *array, of* - зарезервированные слова; *<сп.инд.типов>* - список из одного или нескольких индексных типов, разделенных запятыми; *<тип>* - любой тип Паскаля. Пример:

```
var
    a, b: array [1..10] of Real;
```

Задание 9 – ознакомиться с оператором присваивания и составным оператором языка Паскаль.

Исполняемая часть программы состоит из одного и более ***операторов***. Оператор описывает некоторое действие, которое должна выполнить программа. Подавляющее большинство операторов содержит зарезервированные слова, поясняющее соответствующее действие.

Оператор присваивания – один из наиболее часто используемых операторов обработки данных. Выглядит следующим образом *переменная:=выражение*. И переменная и выражение должны быть одного и того же типа.

Составной оператор – это группа операторов ограниченная зарезервированными словами *begin* (в начале группы) и *end* (в конце). В данном случае эти зарезервированные слова называются ***операторными скобками***. Например:

```
begin
    a:=b+1;
    c:=a*b;
end;
```

В составных операторах можно размещать вложенные составные операторы:

```
begin {Начало основного оператора}
    . . . . .
    begin {Начало вложенного оператора}
        . . . . .
    end; {Конец вложенного оператора}
    . . . . .
end; {Конец основного оператора}
```

Задание 10 – ознакомиться с операторами ввода-вывода языка Паскаль, а также форматом числа.

Для вывода информации на экран служит процедура *Write* и ее модификация *WriteLn*. Формат записи

```
Write('Сообщ1','Сообщ2',ИмяПерем1,Выражение1,...);
WriteLn('Сообщ1','Сообщ2',ИмяПерем1,Выражение1,...);
```

WriteLn отличается от *Write* тем, что после вывода на экран всех своих аргументов переводит курсор на следующую строку, и следующий вывод на экран будет осуществлён с начала следующей строки.

В списке вывода процедур *Write* и *WriteLn* могут встречаться сообщения, заключенные в апострофы (выводятся на экран без изменения), имена переменных (выводится значение переменной), выражения (выводятся значения выражений). Примеры:

```
WriteLn('Дискриминант квадратного уравнения равен',b*b-4*a*c);
WriteLn('x1=',x1,' x2=',x2);
```

Допускается использование *WriteLn* без аргументов. Тогда вывода на экран не происходит, а курсор переводиться на следующую строку.

Для более удобного представления вещественных чисел на экране, есть возможность задавать выводимое количество знаков целой и дробной частей используя ***формат числа***:

WriteLn('Дискриминант квадратного уравнения равен',b:7:3);

где, 7 – это количество позиций отводимое под число, 3 – количество позиций для дробной части. Одну позицию всегда занимает запятая.

Для ввода значений переменных с клавиатуры служит процедура *Read* и её модификация *ReadLn*. Формат записи

Read(ИмяПерем1, ИмяПерем2,…);
ReadLn(ИмяПерем1, ИмяПерем2,…);

Отличие этих двух процедур проявляются при работе с файлами, при вводе с клавиатуры всегда используется *ReadLn*. Пример

ReadLn(a, b, c, d);

Допускается использование *ReadLn()* без аргументов для остановки работы программы до нажатия пользователем клавиши Enter.

2.2. Лабораторные задания

Задание 1 – Укажите, какие из перечисленных идентификаторов содержат ошибки и поясните их

a, 1Program, ALPHA, block#1, MyProgramIsBestProgram,

Date_27_sep_39, My Prog, _beta, mod

Задание 2 – Укажите, какие из перечисленных операторов ввода-вывода содержат ошибки и поясните их

READ M,X, N	*WRITE (6,M) X, X1, X2*
READ (5,6) T, H, M	*WRITE (6,25)*
READ (A:3:5)	*WRITE (B:7:2)*
READ (6,5) B,C	*WRITE 45, F8, 4*

Задание 3 – При помощи программы на языке Паскаль, организуйте вывод значений функции $Y=(X^2)$ в следующем виде

X|	-3|	-2|	-1|	-0|	1|	2|	3|
Y|	|	|	|	|	|	|	|

Задание 4 – Напишите диалоговую программу на языке Паскаль, которая рассчитает значений функции $Y=a*X^2+b$. Диалог должен иметь следующий вид

Начало расчета функции $Y=a*X^2+b$
Введите значение a - _
Введите значение b - _
Введите значение x - _

Для значений a=... b=... x=... значение y=...

Расчет закончен.

2.3. Контрольные вопросы для отчета работы

1. Дайте определение понятия «алгоритм».
2. Назовите основные свойства алгоритма.
3. Изобразите основные виды функциональных блоков при

графическом представлении алгоритма.

4. Что такое язык программирования, синтаксис языка программирования и семантика языка программирования?

5. Приведите классификацию языков программирования.

6. Что такое транслятор, компилятор и интерпретатор?

7. Поясните схему процесса разработки программ в Турбо Паскале?

8. Опишите структура программы на языке Паскаль.

9. Что Вы знаете о константах, выражениях и операциях языка Паскаль?

10. Расскажите об основных типах данных языка Паскаль.

11. Что такое оператор присваивания и составной оператор?

12. Какие операторы предназначены для организации ввода-вывода информации? Поясните их работу на примере.

13. Что такое формат числа? Поясните его работу на примере.

2. ЛАБОРАТОРНАЯ РАБОТА №4

УСЛОВНЫЙ ОПЕРАТОР. ОПЕРАТОР ВЫБОРА. ЦИКЛИЧЕСКИЕ ОПЕРАТОРЫ. РАБОТА С МАССИВАМИ.

Цель работы: закрепить знания об условном операторе, операторе выбора и циклических операторах языка Паскаль. Получить навык работы с одномерными и двухмерными массивами.

Время работы: 8 часов.

2.1. Домашние задания и методические указания по их выполнению

Задание 1 – расширить и закрепить знания о массивах.

Массив – это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его компонентой (или элементом массива).

Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент. Вообще, ***массив это однородный, упорядоченный структурированный тип данных с прямым доступом к элементам***.

Переменные, представляющие компоненты массивов, называются переменными с индексами в отличие от простых переменных, представляющих в программе элементарные данные. Индекс в обозначении компонент массивов может быть константой, переменной или выражением порядкового типа (целочисленный, логический, символьный, перечислимый, диапазон).

Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется линейным. Вообще

количество индексов элементов массива определяет ***размерность массива***. По этом признаку массивы делятся на ***одномерные (линейные), двумерные, трёхмерные*** и т.д.

Например, числовая последовательность четных натуральных чисел 2, 4, 6, ..., N представляет собой линейный массив, элементы которого можно обозначить A[1]=2, A[2]=4, A[3]=6, ..., A[K]=2*(K+1), где K – номер элемента, а 2, 4, 6, ..., N – значения. Индекс (порядковый номер элемента) записывается в квадратных скобках после имени массива.

Например, A[7] – седьмой элемент массива A; D[6] – шестой элемент массива D.

Для размещения массива в памяти ЭВМ отводится поле памяти, размер которого определяется типом, длиной и количеством компонент массива. В языке Pascal эта информация задается в разделе описаний. Массив описывается так:

имя массива: array [тип индекса] of базовый тип;

Чаще всего типом индекса является диапазон. Например,

```
var
B: Array [1..5] Of Real;
R: Array [1..34] Of Char;
```

Заполнить массив можно с помощью оператора присваивания. Этот способ заполнения элементов массива особенно удобен, когда между элементами существует какая-либо зависимость, например, арифметическая или геометрическая прогрессии, или элементы связаны между собой каким-либо соотношением. Другой вариант присваивания значений элементам массива – заполнение значениями, полученными с помощью генератора случайных чисел.

При решении практических задач часто приходится иметь дело с различными таблицами данных, математическим эквивалентом которых служат матрицы. Такой способ организации данных, при котором каждый элемент определяется номером строки и номером

столбца, на пересечении которых он расположен, называется ***двумерным массивом*** или таблицей.

Например, данные о планетах Солнечной системы представлены следующей таблицей:

Таблица 3

Планета	Расстояние до Солнца	Относительный обьем	Относителная масса
Меркурий	57,9	0,06	0,05
Венера	108,2	0,92	0,81
Земля	149,6	1,00	1,00
Марс	227,9	0,15	0,11
Юпитер	978,3	1345,00	318,40
Сатурн	1429,3	767,00	95,20

Их можно занести в память компьютера, используя понятие двумерного массива. Положение элемента в массиве определяется двумя индексами. Они показывают номер строки и номер столбца. Индексы разделяются запятой. Например: A[7, 6], D[56, 47].

В разделе описаний двухмерный массив описывается, например, следующим образом:

```
var
M: array [1..5,1..10] of integer;
```

Заполняется двумерный массив аналогично одномерному, например, A[1,1]=457; A[1,2]=342,5; A[2,1]=458; A[2,2]=33; A[3,1]=0; A[3,2]=-234

Задание 2 – закрепить знания об условном операторе ***if*** языка Паскаль.

Условный оператор ***if*** может иметь две формы – краткую и полную.

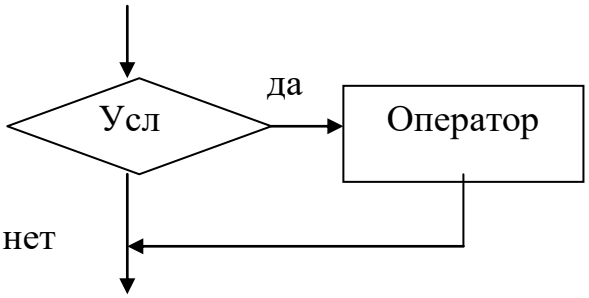

Рис. 4. Алгоритмическая схема краткой формы условного оператора

Эта схема соответствует такой строке программного кода

if *Усл* ***then*** *Оператор;*

При выполнении оператора анализируется условие ***Усл***. Если оно имеет значение ***True***, выполняется *Оператор*, иначе ничего не происходит и условный оператор завершает свою работу. Оператор может быть составным, т.е. определять группу операторов, которые будут выполнятся или пропускаться в зависимости от условия. Программный код в этом случае имеет вид

```
if Усл then
begin
    Оператор1;
    Оператор2;
    Оператор3;
end;
```

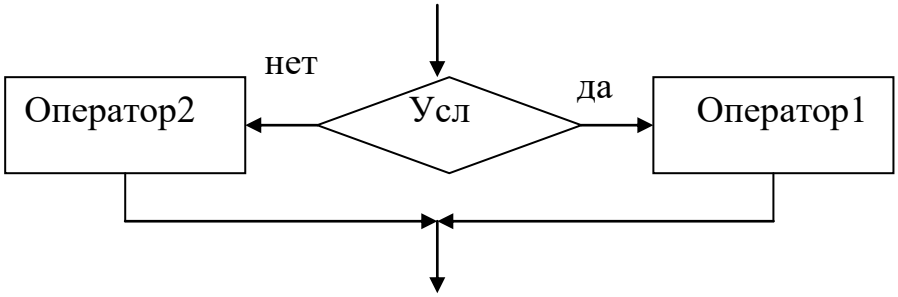

Рис. 5. Алгоритмическая схема полной формы условного оператора

Эта схема соответствует такой строке программного кода

if *Усл* ***then*** *Оператор1* ***else*** *Оператор2;*

При выполнении оператора анализируется условие *Усл*. Если оно имеет значение ***True***, выполняется *Оператор1*, иначе выполняется *Оператор2* оператор завершает свою работу. Как и в краткой форме *Оператор1* и *Оператор2* могут быть составными.

```
if Усл then
begin
    Оператор1;
    Оператор2;
    Оператор3;
end;
else
begin
    Оператор4;
    Оператор5;
    Оператор6;
end;
```

Задание 3 – закрепить знания об операторе выбора ***case*** языка Паскаль.

Оператор выбора ***case*** позволяет выбрать одно из нескольких

возможных продолжений программы. Параметром, по которому осуществляется выбор, служит *ключ выбора* – выражение любого порядкового типа (целого, символьного).

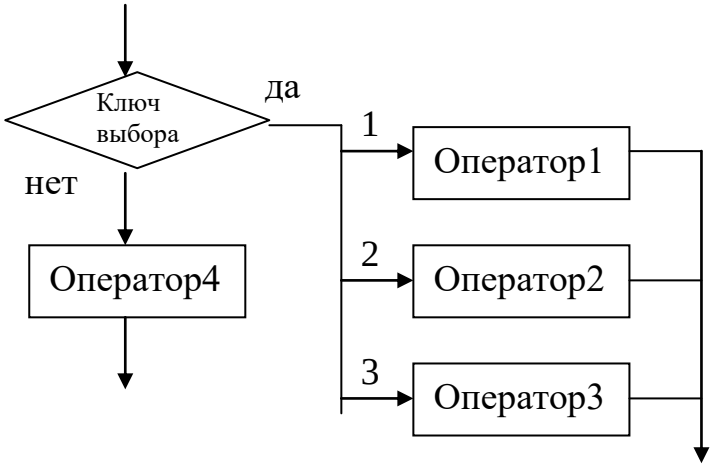

Рис. 6. Алгоритмическая схема оператора выбора

Эта схема соответствует такой строке программного кода

case <*ключ_выбора*> ***of*** <*список_выбора*> ***else*** <*оператор*> ***end;***

Оператор работает следующим образом, вначале вычисляется выражение <*ключ_выбора*>, а затем в последовательности операторов <*список_выбора*> выполняется тот, которому предшествует константа, равная вычисленному значению, после чего оператор выбора завершает свою работу. Если в списке выбора не будет найдена, соответствующая вычисленному значению ключа выбора, управление передается операторам, стоящим за словом ***else***. Конструкцию ***else*** можно не указывать, тогда оператор выбора просто завершит свою работу. Любому из операторов списка выбора может предшествовать не одна, а несколько констант выбора, разделенных запятыми.

Задание 4 – закрепить знания о счетном операторе цикла ***for*** языка Паскаль.

Циклические операторы используются для многократного повторения некоторого фрагмента программы, который называют

телом цикла. Если количество повторений заранее известно, то используется счетный оператор цикла ***for***

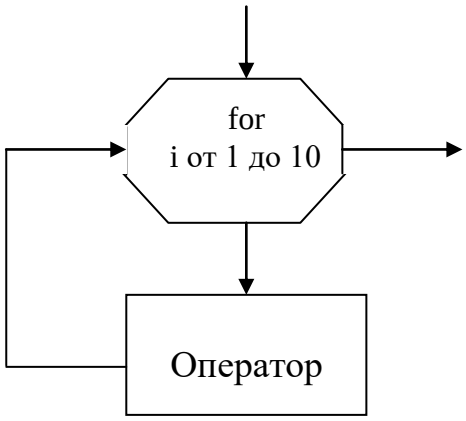

Рис. 7. Алгоритмическая схема счетного оператора цикла

Эта схема соответствует такой строке программного кода

for *<параметр_цикла>:=<нач_значение>* ***to*** *<кон_значение>* ***do*** *Оператор*

При выполнении оператора ***for*** вначале вычисляется значение *<нач_значение>* и осуществляется присваивание *<параметр_цикла>:=<нач_значение>*. После этого циклически повторяется следующая последовательность действий:

1. Проверяется условие *<параметр_цикла>* > *<кон_значение>*. Если условие выполнено, оператор ***for*** прекращает свою работу, иначе п.2

2. Выполняется *Оператор*

3. Происходит наращивание переменной *<параметр_цикла>* на единицу

Задание 5 – закрепить знания о цикле с предпроверкой условия ***while*** языка Паскаль.

Оператор цикла с предварительной проверкой условия ***while*** в алгоритмической форме имеет вид

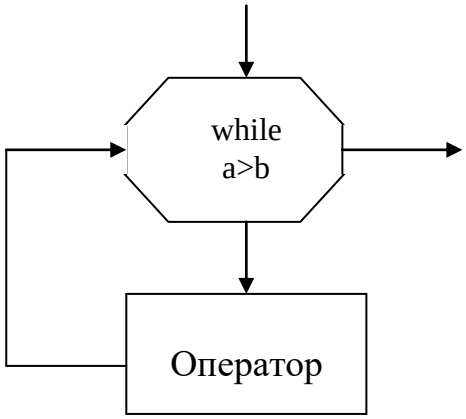

Рис. 8. Алгоритмическая схема цикла с предпроверкой условия

и записывается следующим образом

while *Усл* ***do*** *Оператор;*

При выполнении оператора анализируется условие *Усл*. Если оно имеет значение ***True***, выполняется *Оператор*, после чего вычисление выражения *Усл* и его проверка повторяются. Если *Усл* имеет значение ***False***, оператор ***while*** прекращает свою работу.

В случае если в теле цикла необходимо выполнять несколько операторов, используются операторные скобки ***begin-end*** между которыми помещается тело цикла.

```
while Усл do
begin
    Оператор1;
    Оператор2;
    Оператор3;
end;
```

Задание 6 – закрепить знания о цикле с постпроверкой условия ***repeat … until*** языка Паскаль.

Оператор цикла с постпроверкой проверкой условия ***repeat … until*** в алгоритмической форме имеет вид

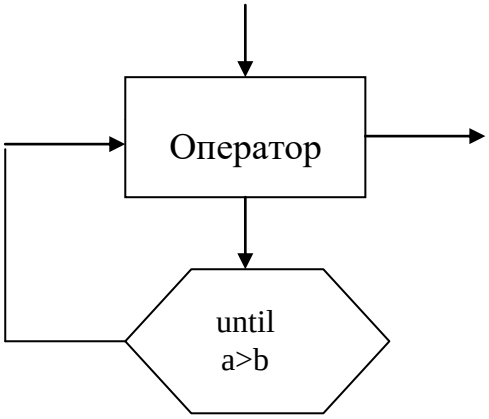

Рис. 9. Алгоритмическая схема цикла с постпроверкой условия

и записывается следующим образом
repeat *Оператор* ***until*** *Усл;*

Оператор выполняется хотя бы один раз, после чего вычисляется выражение *Усл*. Если его значение ***False***, то *Оператор* повторяется, в противном случае цикл ***repeat … until*** завершает свою работу.

Задание 7 – разобрать и уяснить алгоритм и программу на языке Паскаль, в которой пользователь заполняет одномерный массив содержащий 10 элементов, а затем он выводится на экран.

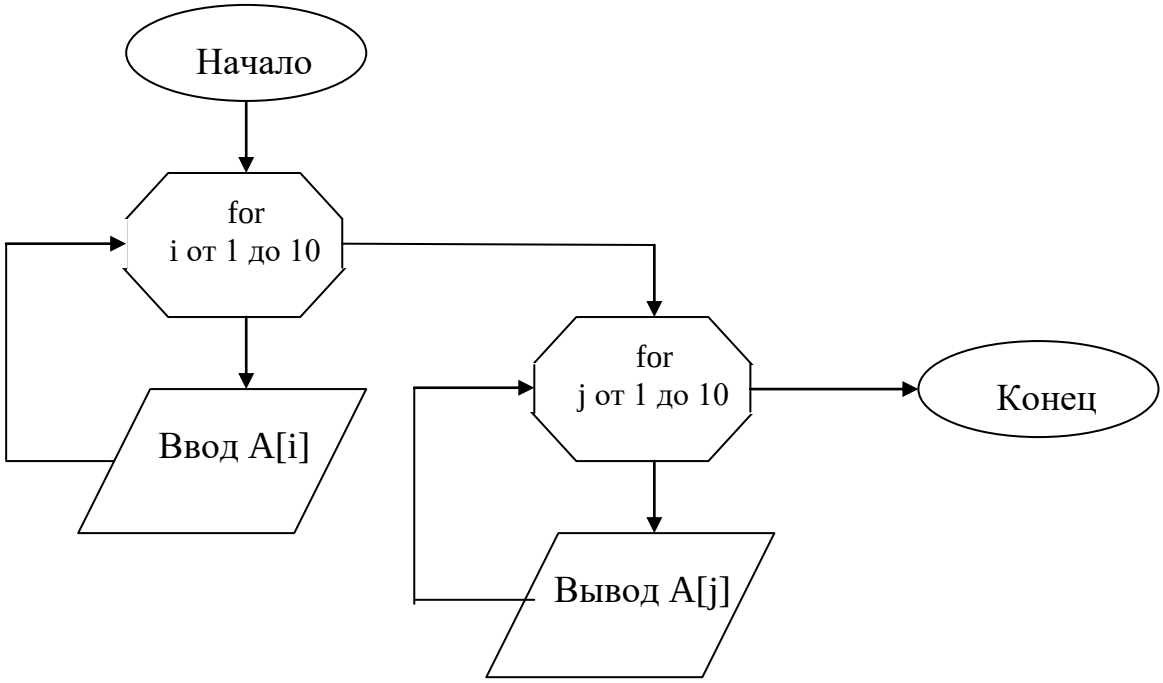

Рис. 10

```
Program test1;
var
      A: array [1..10] of real;
      i, j: integer;
begin
      for i:=1 to 10 do
      begin
      writeln(`Введите значение A[`,I,`]`);
readln(A[i]);
            end;
      writeln(`Вывод массива A`);
      for j:=1 to 10 do
      begin
      writeln(`A[`,J,`]=`,A[j]:7:3);
            end;
end.
```

Задание 8 – разобрать и уяснить алгоритм и программу на языке Паскаль, в которой случайным образом заполняется двухмерный массив размером 5x6 элементов, а затем он выводится на экран.

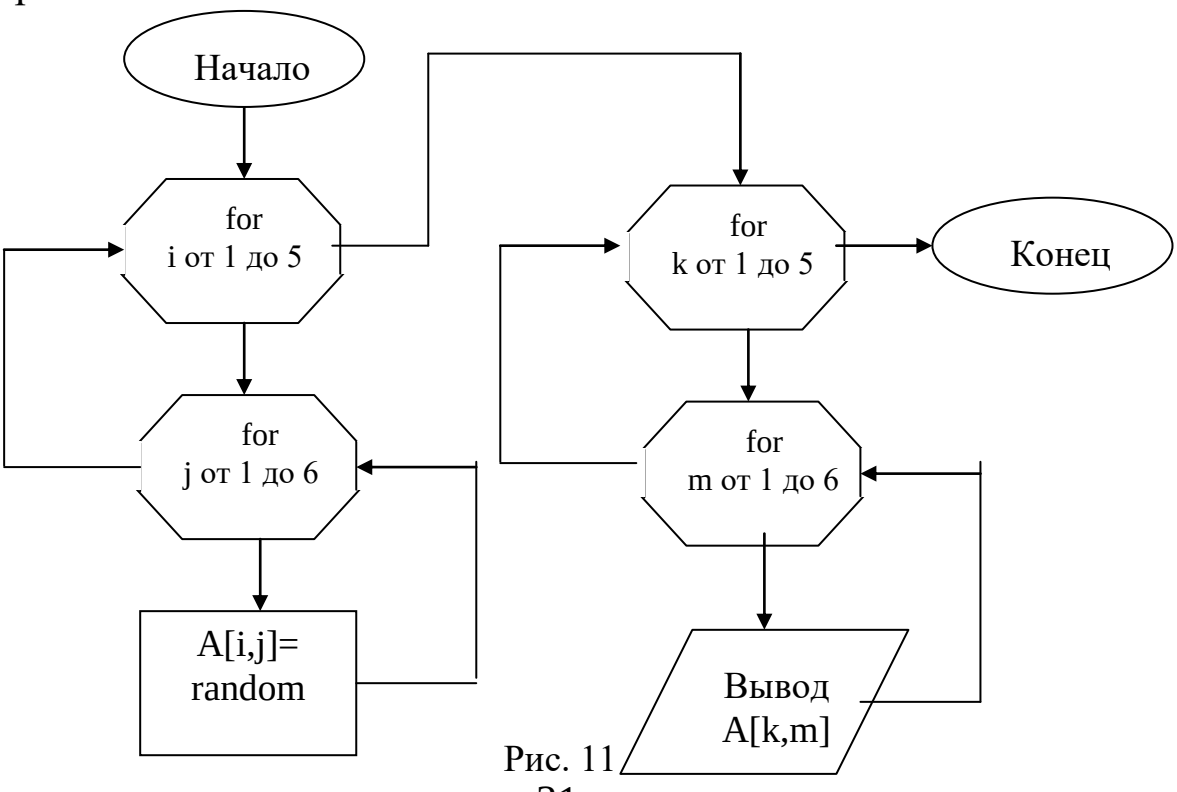

Рис. 11

```
Program test2;
var
    A: array [1..5,1..6] of real;
    i, j, k, m: integer;
begin
    for i:=1 to 5 do
    begin
    for j:=1 to 6 do
    A[i,j]:=random(100);
        end;
    writeln(`Вывод массива A`);
    for k:=1 to 5 do
    begin
    for k:=1 to 6 do
        writeln(`A[`,K,`,`,M,`]=`,A[k,m]:7:3);
        end;
end.
```

2.2. Лабораторные задания

Задание 1 – Составьте блок-схему алгоритма и напишите программу на языке Паскаль для работы с одномерным массивом в соответствии со своим вариантом. Массив должен заполняться пользователем вручную.

1. дан одномерный массив, сдвинуть циклически его элементы влево на три позиции
2. дан одномерный массив, сдвинуть циклически его элементы вправо на две позиции
3. дан одномерный массив, заменить каждый элемент на сумму стоящих после него элементов
4. дан одномерный массив, заменить элементы, имеющие дробную часть на нули
5. дан одномерный массив, упорядочить элементы в нем по возрастанию

6. дан одномерный массив, упорядочить элементы в нем по убыванию

7. дан одномерный массив, найти в нем максимальный элемент

8. дан одномерный массив, найти в нем минимальный элемент

9. дан одномерный массив, найти в нем порядковый номер максимального элемента

10. дан одномерный массив, найти в нем порядковый номер минимального элемента

Задание 2 – Составьте блок-схему алгоритма и напишите программу на языке Паскаль для работы с двухмерным массивом в соответствии со своим вариантом. Массив должен заполняться автоматически случайными числами.

1. дана матрица порядка M × M, найти произведение элементов, лежащих на главной диагонали

2. дана матрица порядка M × M, найти первый отрицательный и первый положительный элемент

3. дана матрица порядка M × M, найти индексы первого отрицательного и первого положительного элемента

4. дана матрица порядка M × M, заменить ее элементы на минимальный элемент

5. дана матрица порядка M × M, найти произведение элементов в каждой строке

6. дана матрица порядка M × M, выполнить ее транспонирование (замена строк на столбцы)

7. дана матрица порядка M × M, найти среднее арифметическое элементов в каждой строке

8. дана матрица порядка M × M, найти среднее арифметическое элементов в каждом столбце

9. дана матрица порядка M × M, найти максимальный элемент, лежащий ниже главной диагонали

10. даны две матрицы порядка M × M, найти их поэлементное произведение

2.3 Контрольные вопросы для отчета работы

1. Что такое массив? Почему массив является структурированным типом данных?
2. Что такое размерность массива? Какими способами может быть заполнен массив?
3. Расскажите об условном операторе ***if*** языка Паскаль.
4. Расскажите об операторе выбора ***case*** языка Паскаль.
5. Расскажите о счетном операторе цикла ***for*** языка Паскаль.
6. Расскажите о цикле с предпроверкой условия ***while*** языка Паскаль.
7. Расскажите о цикле с постпроверкой условия ***repeat … until*** языка Паскаль.
8. Изобразите и поясните алгоритм программы, в которой пользователь заполняет одномерный массив, содержащий 8 элементов, а затем он выводится на экран.
9. Напишите и поясните программу на языке Паскаль, в которой случайным образом заполняется одномерный массив, содержащий 15 элементов, а затем он выводится на экран.
10. Изобразите и поясните алгоритм программы, в которой случайным образом заполняется двухмерный массив размером 7x8 элементов, а затем он выводится на экран.
11. Напишите и поясните программу на языке Паскаль, в которой пользователь заполняет двухмерный массив размером 4x5 элементов, а затем он выводится на экран.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Фаронов В.В. Турбо Паскаль 7.0: Начальный курс: учебное пособие / В.В. Фаронов. – 7-е изд., перераб. – М.: Нолидж, 2002. – 576 с.: ил.

2. Фаронов В.В. Турбо Паскаль: учебное пособие / В.В. Фаронов. – СПб.: Питер, 2007. – 367 с.: ил.

3. Информатика: Базовый курс: учеб. пособие для втузов / под ред. С.В. Симоновича. СПб.: Питер, 2003. – 640 с.: ил.

4. Архангельский А.Я. Программирование в Delphi: учебник по классическим версиям Delphi / А.Я. Архангельский. – М.: Бином, 2006. – 1152 с.: ил.

СОДЕРЖАНИЕ

1. Лабораторная работа №3 1
2. Лабораторная работа №4 20
Библиографический список 33

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ № 3-4 по дисциплине «Информатика» для студентов направления 12.03.01 «Приборостроение» (профиль «Приборостроение») всех форм обучения

Составители:
д-р. техн. наук М.А. Ромащенко,
канд. техн. наук А.А. Пирогов,
И.В. Свиридова

Компьютерный набор М.А. Ромащенко

Подписано к изданию _____
Уч.-изд. л. _____

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский проспект, 14