Уважаемые студенты групп!

Вашему вниманию представлена лекция на тему «История создания языка Паскаль. Алфавит языка. Типы величин. Основные типы данных. Структура программы». Лекция рассчитана на 8 часов (на 4 пары)

Задание

1. Прочитать внимательно лекцию.
2. Законспектировать лекцию в рабочую тетрадь не менее 3-5 страницы рукописного текста. В конспекте лекции обязательно должно быть приведены примеры.
3. Решить приведенные в лекции в контрольных вопросах задачи.
4. Дата предоставления полного фотоотчета лекции будет сообщена дополнительно.

С уважением Ганзенко Ирина Владимировна

!!! Если возникнут вопросы обращаться по телефону 0721134803 (вацап), +79591134803 (телеграмм)

disobuch.ganzenko2020@mail.ru

Лекция 2. История создания языка Паскаль. Алфавит языка. Типы величин. Основные типы данных. Структура программы

План

1. Свойства алгоритма.
2 Виды алгоритмов. Представление алгоритмов в виде блок-схемы
3 Алфавит языка Турбо Паскаль
4 Классификация типов данных
5. Стандартные функции.
6. Выражения.
7 Структура программы на языке Паскаль, операторы языка
8. Форматированный вывод
9. Составленный оператор
10 Контрольные вопросы и задания для самостоятельной работы

Понятие алгоритма так же фундаментально для информатики, как и понятие информации.

Само слово «алгоритм» происходит от имени выдающегося математика средневекового Востока Мухаммеда аль-Хорезми. Им были предложены приемы выполнения арифметических вычислений с многозначными числами

(они всем хорошо знакомы из школьной математики). Позже в Европе эти приемы назвали алгоритмами от «Algorithmi» — латинского написания имени аль-Хорезми. В наше время понятие алгоритма понимается шире, не ограничиваясь только арифметическими вычислениями.

Термин «алгоритм» стал достаточно распространенным не только в информатике, но и в быту. Под алгоритмом понимают описание какой-либо последовательности действий для достижения заданной цели. В этом смысле, например, алгоритмами можно назвать инструкцию по использованию кухонного комбайна, кулинарный рецепт, правила перехода улицы и пр.

Для использования понятия алгоритма в информатике требуется более точное определение, чем данное выше. Получим такое определение.

Ключевыми словами, раскрывающими смысл этого понятия, являются: исполнитель, команда, система команд исполнителя.

Алгоритм представляет из себя последовательность команд определяющих действия *исполнителя* (субъекта или управляемого объекта). Всякий алгоритм составляется в расчете на конкретного исполнителя с учетом его возможностей. Для того чтобы алгоритм был выполним, нельзя включать в него команды, которые исполнитель не в состоянии выполнить. Нельзя повару поручать работу токаря, какая бы подробная инструкция ему не давалась. У каждого исполнителя имеется свой перечень команд, которые он может исполнить. (Такой перечень называется *системой команд исполнителя алгоритмов* (СКИ)).

1. Свойства алгоритма.

Дискретность.

Процесс решения задачи должен быть разбит на последовательность отдельных шагов. Таким образом, формируется упорядоченная совокупность отделенных друг от друга команд (предписаний). Образующаяся структура алгоритма оказывается прерывной (дискретной): только выполнив одну команду, исполнитель сможет приступить к выполнению следующей.

Точность (определенность).

Каждая команда алгоритма должна определять однозначное действие исполнителя. Это требование называется точностью алгоритма.

Понятность.

Алгоритм, составленный для конкретного исполнителя, должен включать только те команды, которые входят в его систему команд. Это свойство алгоритма называется понятностью. Алгоритм не должен быть рассчитан на принятие каких-либо самостоятельных решений исполнителем, не предусмотренных составителем алгоритма.

Конечность (результативность).

Еще одно важное требование, предъявляемое к алгоритму, —это конечность (иногда говорят — результативность) алгоритма. Это значит, что исполнение алгоритма должно завершиться за конечное число шагов.

Массовость.

Разработка алгоритмов — процесс интересный, творческий, но непростой, требующий многих умственных усилий и затрат времени. Поэтому предпочтительно разрабатывать алгоритмы, обеспечивающие решение всего класса задач данного типа. Например, если составляется алгоритм решения квадратного уравнения $AX^2 + BX + C = O$, то он должен быть вариативен, т. е. обеспечивать возможность решения для любых допустимых исходных значений коэффициентов *A, B, C.* Про такой алгоритм говорят, что он удовлетворяет требованию массовости.

Свойство массовости не является необходимым свойством алгоритма. Оно скорее определяет качество алгоритма; в то же время свойства точности, понятности и конечности являются необходимыми (иначе это не алгоритм).

Для успешного выполнения любой работы мало иметь ее алгоритм. Всегда требуются еще какие-то *исходные данные*, с которыми будет работать исполнитель (продукты для приготовления блюда, детали для сбора технического устройства и т. п.). Исполнителю, решающему математическую задачу, требуется исходная числовая информация. Задача всегда формулируется так: дана *исходная информация, требуется получить какой-то результат.* В математике вы привыкли в таком виде записывать условия задач. Например:

Дано: катеты прямоугольного треугольника a=3 см; b=4 см.

Найти: гипотенузу c.

Приступая к решению любой задачи, нужно сначала собрать все необходимые для ее решения данные.

Еще *пример*: для поиска номера телефона нужного вам человека исходными данными являются: фамилия, инициалы человека и телефонная книга (точнее, информация, заключенная в телефонную книгу). Однако этого может оказаться недостаточно. Например, вы ищите телефон А. И. Смирнова и обнаруживаете, что в книге пять строк с фамилией «Смирнов А. И.» Ваши исходные данные оказались *неполными* для точного решения задачи (вместо одного телефона вы получили пять). Оказалось, что нужно знать еще домашний адрес. Набор: фамилия — инициалы — телефонный справочник — адрес — является *полным набором данных в* этой ситуации. Только имея полный набор данных, можно точно решить задачу. Обобщая все сказанное, сформулируем определение алгоритма.

Алгоритм — понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

Если алгоритм обладает перечисленными выше свойствами, то работа по нему будет производиться исполнителем *формально* (т. е. без всяких элементов творчества с его стороны). На этом основана работа программно-управляемых исполнителей-автоматов, например промышленных роботов. Робот-манипулятор может выполнять работу токаря, если он умеет делать все операции токаря (включать станок, закреплять резец, перемещать резец, замерять изделие и т. д.). От исполнителя не требуется понимание сущности

алгоритма, он должен лишь точно выполнять команды, не нарушая их последовательности.

А что такое программа? Отличается ли чем-то программа от алгоритма? Программа — это алгоритм, записанный на языке исполнителя. Иначе можно сказать так: алгоритм и программа не отличаются по содержанию, но могут отличаться по форме.

Для алгоритма строго не определяется форма его представления. Алгоритм можно изобразить графически (блок-схемы), можно — словесно, можно — какими-нибудь специальными значками, понятными только его автору. Но программа должна быть записана на языке исполнителя (для ЭВМ это язык программирования).

4 Виды алгоритмов. Представление алгоритмов в виде блок-схемы

Алгоритмы бывают разные. Самый простой линейный алгоритм.

Линейный (последовательный) алгоритм – описание действий, которые выполняются однократно в заданном порядке.

Пример: алгоритм решение задачи (от записи данных до ответа), алгоритм открывания двери (вставить ключ, повернуть ключ, открыть дверь) и т. д.

Разветвляющий алгоритм – алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Условие – выражение, находящееся между словом «если» и словом «то» и принимающее значение «истина» или «ложь».

Пример: алгоритм нахождения функции не определенной на всей числовой прямой (находим значение *y* по заданному значению *x*, *если x* определена в этой точке), алгоритм покупки билетов (спрашиваем в кассе, есть ли билеты, *если* билеты есть, *то* подаем деньги, получаем билеты) и т. д.

Циклический алгоритм – описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Перечень повторяющихся действий называется телом цикла.

Пример: алгоритм нахождения значений *y* при заданных или задающихся значениях *x* для построения графика функции (находятся значения на определенном интервале с заданным шагом), алгоритм покраски забора (макнуть кисть в краску, покрасить доску, шаг влево, …).

Вспомогательный алгоритм – алгоритм, который можно использовать в других алгоритмах, указав только его имя.

Вспомогательному алгоритму должно быть присвоено имя.

Стандартные графические объекты блок-схемы представлены в таблице 2.1

Таблица 2.1

Стандартные графические объекты блок-схемы

Вид стандартного графического объекта	**Назначение**
начало	Начало алгоритма
конец	Конец алгоритма
дествие	Выполняемое действие (расчет по формуле)
ввод	Ввод данных
выво	Вывод данных
?	Условие выполнения действия
Подпрогра ммаграмма	Вспомогательный алгоритм
	Последовательность выполнения действий

3. Алфавит языка Турбо Паскаль

Алфавит языка Турбо Паскаль включает буквы, цифры, шестнадцатеричные цифры, специальные символы и зарезервированные слова.

1) Заглавные и строчные буквы латинского алфавита:
A, B, … X, Y, Z, a, b, … x, y, z и знак подчёркивания _.
2) Арабские цифры: 0, 1, …, 9.
3) Шестнадцатеричные цифры: 0, 1, …, 9, A, B, …, F.
4) Специальные символы (22 символа):
+ - * / = > < . , ; : @ ‘ ( ) [ ] { } # $ ^
5) Зарезервированные слова:
Begin, end, case, const …

В Турбо Паскале определены следующие операции:

1) арифметические +, –, *, /, div, mod
2) отношения =, <>, <, >, <=, >=

3) логические not, or, and, shl, shr, xor

4) над множествами * (пересечение), + (сложение или объединение), – (разность), in (принадлежность).

Идентификаторы (имена) в Турбо Паскале – это имена констант, переменных, типов, меток, объектов, процедур, функций, модулей, программ, полей в записях, файлов. Идентификаторы могут иметь произвольную длину, но значащими являются только первые 63 символа. Идентификатор должен начинаться буквой или знаком подчёркивания, за которым могут следовать буквы, цифры, знак подчёркивания. Пробелы и специальные символы не могут входить в идентификатор. В качестве идентификаторов нельзя использовать зарезервированные слова. Прописные и строчные латинские буквы воспринимаются равнозначно, т.е. indmax и INDMAX будут восприняты одинаково.

Примеры правильных идентификаторов:

A1, mas_3, _max, alfa

Примеры неправильных идентификаторов:

1_a, block#1, my primer, begin

в) Решение задачи на компьютере – это процесс сбора, обработки и передачи информации. Поэтому любая программа имеет смысл, если она обрабатывает какие-либо данные. Как и в других языках программирования данные разделяются на константы и переменные.

Константами называются элементы данных, значения которых установлены в описательной части программы и не изменяются в процессе её выполнения. Константы могут быть целого, вещественного, символьного, логического, строкового типа. Используются они в программе в явной форме или с помощью идентификатора константы. **Переменными** называются величины, значения которых могут изменяться в процессе выполнения программы. Все используемые в программе переменные должны быть описаны с указанием их типов. Имена переменных желательно выбирать таким образом, чтобы было понятно их назначение.

4 Классификация типов данных

Данные – это общее понятие для всего того, чем оперирует компьютер. Под типом данных следует понимать множество значений, которые может принимать переменная, и те операции, которые можно к этим значениям применять.

В языке Турбо Паскаль используются следующие типы данных:

⇒ простые типы;

⇒ структурированные типы;

⇒ указатели;

⇒ процедурные типы;

⇒ объекты.

Среди простых типов выделяют стандартные (предопределённые): целый, вещественный, логический, символьный и нестандартные (определяемые программистом): перечисляемый, диапазонный (ограниченный).

Стандартные простые типы данных.

Целые типы. Имеется пять целых типов данных, различающихся диапазоном допустимых значений и размером памяти, отводимой для их представления.

Тип	Диапазон значений	Объем памяти
ShortInt	-128 ... 127	1 байт со знаковым битом
Integer	-32 768 … 32 767	2 байта со знаковым битом
Longlnt	-2 147 483 648 ... 2 147 483 647	4 байта со знаковым битом
Byte	0… 255	1 байт без знакового бита
Word	0 ... 65 535	2 байта без знакового бита

Вещественные типы. Имеется пять стандартных вещественных типов, характеристики которых представлены в таблице

Вещественные типы		
Размер в байтах	Название типа	Диапазон
6	real	-2.9^{39}…$+1.7^{38}$
4	single	-1.5^{45}…$+3.4^{38}$
8	double	-5.0^{324}…$+1.5^{308}$
10	extended	-3.4^{4951}…$+1.1^{4932}$
8	comp	$-2^{63}+1$…$+2^{63}-1$

Логический тип.

Переменные логического типа описываются с помощью идентификатора Boolean. Они могут принимать два значения: *False* (ложь) или *True* (истина) (*False* и *True* — стандартные константы), под них выделяется 1 байт памяти.

В Турбо Паскале имеются три логические операции: **или** — *or*; **и** — *and*; **отрицание** — *not*.

Следует четко понимать, что результатом выполнения операций сравнения (отношения): < (меньше), > (больше), <= (меньше или решаю), >= (больше или равно), <> (не равно), = (равно) является величина логического типа. Ее значение равно *True*, если отношение выполняется для значений входящих в него операндов, и *False* — в противном случае.

Символьный тип.

Символьный тип char определяется множеством значений кодовой таблицы. Каждому символу приписывается целое число в диапазоне от 0 до 255. Для кодировки используется код **ASCII.** Для размещения в памяти переменной символьного типа требуется один байт.

Пользовательские типы.

Кроме стандартных типов данных Паскаль поддерживает скалярные типы, определяемые самим пользователем. К ним относятся перечисляемый и интервальный типы. Данные этих типов занимают в памяти один байт, поэтому скалярные пользовательские типы не могут содержать более 256 элементов.

Объявление перечисляемого типа описывает множество значений, которые может принимать переменная заданного типа.

<имя типа>=(значение1, значение2, …, значениеN);

Интервальный тип.

Этот тип задаётся двумя константами, определяющими границы диапазона значений для данной переменной. При каждой операции с переменной интервального типа проверяется соответствие её значения заданному диапазону.

<имя типа>=константа1**..** константа2;

5. Стандартные функции.

Обращение	Функция
Abs(x)	Модуль аргумента
Frac(x)	Дробная часть числа
Int(x)	Целая часть числа
Round(x)	Округление до ближайшего целого
Sqr(x)	Квадрат числа
Sqrt(x)	Корень квадратный
Trunc(x)	Ближайшее целое, не превышающее x по модулю
Exp(x)	экспонента
Ln(x)	натуральный логарифм, x>0
sin(x)	синус, угол в радианах
cos(x)	косинус, угол в радианах
arctan(x)	арктангенс
Pi	Число π=3.141592653…

Odd(x)	true, если x – нечётное, false, если x – чётное
Ord(x)	преобразование порядкового типа в целый тип
Chr(x)	преобразует ASCII-код в символ
Pred(x)	определение предыдущего значения величины x
Succ(x)	определение последующего значения величины x

6. Выражения.

Выражение представляет собой формальное правило для вычисления некоторого значения. Оно формируется из констант, переменных, функций, знаков операций и круглых скобок.

Выражения состоят из операций и операндов. Различают бинарные операции – они выполняются над двумя операндами, и унарные (одноместные) — над одним операндом. Бинарные операций записываются в инфиксной форме (знак операции ставится между операндами), унарные — в префиксной (знак унарной операции предшествует операнду). Порядок выполнения операций в выражении определяется их приоритетами, которые приведены в следующей таблице.

Операции	**Приоритет**	**Тип операции**
@, Not, +, -, ^	1-й (высший)	унарный
*, /, Div, Mod , And, Shl, Shr	2-й	мультипликативный
+, -, Or, Xor	3-й	аддитивный
=, <>, <, >, <=, >=, In	4-й (низший)	операции отношения

Арифметические операции

Выражение	Операция
A+B	Сложение
A–B	Вычитание
A*B	Умножение
A/B	Вещественное деление
A div B	Целое деление
A mod B	Остаток от целого деления

Примеры выражений на языке Паскаль:
a/sqr(x); x+y; x>y; 5*(d+sqrt(b)); (x>=a) and (x<=b).

7 **Структура программы на языке Паскаль, операторы языка**

Program <имя программы>;
Label <раздел описания меток>;
Const <раздел описания констант>;

Type <раздел описания типов>;
Var <раздел описания переменных>;
Function (Procedure);
Begin
<раздел операторов>
End.

Операторы языка Паскаль

Простые (другое название - линейные) программы состоят из команд присвоения, ввода-вывода данных и вызова процедур.

Команда присвоения

Команда присваивания имеет вид (синтаксис):

<Имя переменной> = <выражение>;

Действие команды (семантика). Вычисляется выражение и его значение придается переменной. Выражение предназначено для описания формул, по которым выполняются вычисления. Выражение может содержать числа, переменные, название функции, соединенные знаками операций.

Пример.

A = B + C;

ALFA = 6 * D;

Z1 = 12;

Переменная и выражение должны быть одного типа или согласованными: переменным вещественного типа можно придавать значение выражений целого типа, а переменным строчной типа присваивать значения выражений символьного типа, но не наоборот.

Команды ввода (read, readln) данных

Присваивать значения переменным можно двумя способами:

с помощью команды присвоения, например x:=5, или команд ввода данных с клавиатуры. Второй способ делает программу более универсальной, поскольку позволяет решать задачи для различных значений переменных.

Команда **read** имеет вид:

read (<переменная 1>, ..., <переменная n>)

Действие команды. Выполнение программы останавливается. Система переходит в режим ожидания ввода данных. Значение этих данных пользователь набирает на клавиатуре через промежуток или нажимает после каждого данного Enter. В результате выполнения этой команды соответствующим переменным будут присвоены конкретные значения.

Команда **readln** имеет вид:

readln (<переменная 1>, ..., <переменная n>)

Между указаниями read и readln есть разница. После выполнения указания read курсор останется в этой же строке. После выполнения указания readln будет сделано переход в следующую строку.

Замечания. Команду readln без параметров часто используют в среде TP, чтобы осмотреть результаты выполнения программы на экране. Чтобы

после этого перейти в режим редактирования программы, нужно нажать на клавишу ввода (Enter).

Команды вывода (write, writeln) данных

Для вывода на экран сообщений и результатов вычислений используют команды **write** и **writeln:**

write (список);

writeln (список);

В списке вывода перечисляются через запятую переменные, выражения или текстовые константы.

Команда writeln действует так же как и команда write, но после ее выполнения курсор переходит в следующую строку. Таким образом вывод значений следующей команды (write или writeln) будет в новой строке.

пример

23 12 августа {Вывод данных с помощью команды write}

23

12 {Вывод данных с помощью команды writeln}

августа

Для перехода на новую строку экрана или для пропуска строки используют команду writeln без параметров.

Для удобства вывода данных пользуются своеобразными подсказками пользователя (текстовыми константами).

пример

writeln ('X =', X);

writeln ('Y =', Y)

writeln ('Z =', Z);

На экран будет выведено результат:

X = 23

Y = 12

Z = 8

Как видно из примера, текстовые константы ('X =', 'Y =', 'Z =') или подсказки пользователя должны с двух сторон браться в апострофы и отделяться от переменной запятой.

Для удобства ввода данных пользуются сочетанием указаний read и write.

пример

read ('Введите значение X:');

writeln (X)

read ('Введите значение Y:');

writeln (Y)

read ('Введите значение Z:');

writeln (Z)

Общий вид последовательного выполнения данных команд:

Введите значение X: 23

Введите значение Y: 12

Введите значение Z: 8

8. Форматированный вывод

Команды write и writeln могут осуществлять форматный вывод данных. Форматирование - это представление результатов в заранее заданном пользователем виде. Для этого после выражения через двоеточие записывают число (:n) - количество позиций на экране, которые надо предоставить для вывода значения этого выражения. Формат :n применяют для данных целого и строчной типов. Во время вывода данных действительного типа отмечают общее количество позиций для всех символов (n) и количество позиций для дробной части (m), то есть формат имеет вид : n: m.

Пример (A = 12,8)

writeln ('A =', A);

На экране получим следующий результат:

A = 1.2800000000E + 01

writeln ('A =', A: 5: 2)

результат:

A = 12.80

9. Составленный оператор

Составной оператор - это последовательность произвольных команд программы, отделенных друг от друга точкой с запятой, взятых в операторные скобки - служебные (зарезервированные) слова begin ... end.

формат:

```
begin
<Оператор1>;
<Оператор2>;
...
<ОператорN>
end;
```

После служебного слова end ставится точка с запятой. В некоторых случаях, когда составной оператор используется в командах ветвления, точка с запятой может и не записываться.

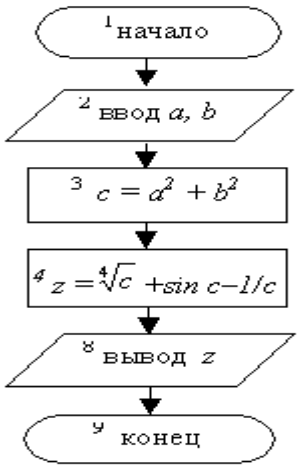
пример

```
begin
a = 3.5;
b = 7.2;
s = a + b;
writeln ( 's =', s)
end;
```

Пример алгоритма и программы линейной структуры

Даны переменные a и b. Найти $Z = \sqrt[4]{a^2 + b^2} + \sin(a^2 + b^2) - \frac{1}{a^2 + b^2}$

При составлении алгоритма необходимо выделить однотипные выражения (здесь a^2+b^2), которые достаточно посчитать один раз, а затем использовать результат вычислений. Желательно разбить сложные вычисления одного выражения на более простые (например, отдельно вычислить числитель и знаменатель дроби) для того, чтобы в алгоритме не было громоздких формул. На рисунке 2.1 представлен вариант блок-схемы алгоритма и программа.




```
Program Primer1;
Var a,b,c,z:real;
Begin
   Write ('Введите a и b');
   Read (a, b);
   C := sqr(a) + sqr(b);
   z := sqrt (sqrt(c)) + sin(c) – 1/c;
   Write('Z=', z:10:3)
End.
```

Рисунок 2.1 - Блок-схема алгоритма и программа линейной структуры

Вычисление значения Z производится в следующей последовательности:

а) в блоке 2 вводятся исходные данные – значения *a* и *b*;

б) в блоке 3 вычисляется арифметическое выражение a^2+b^2, и результат запоминается в переменной *c*;

в) в блоках 4-6 вычисляются первое слагаемое, числитель и знаменатель второго слагаемого;

г) в блоке 7 производится окончательный расчет *Z*;

д) в блоке 8 выводятся исходные данные и результат.

В программе действия блоков 3-7 записываются операторами присваивания, блоки 2 и 8 реализуются операторами ввода/вывода. Ввод осуществляется с запросом, поэтому сначала записан оператор **Write**, а затем **Read**. Вывод осуществляется форматным способом. Все переменные, участвующие в программе, объявляются в разделе **Var** ее описательной части.

10 Контрольные вопросы и задания для самостоятельной работы

1. Перечислите основные этапы решения задач с помощью компьютера.
2. Что такое алгоритм? Каким свойствам он должен удовлетворять?
3. Какие Вы знаете способы записи алгоритмов? Приведите примеры.
4. Классификация языков программирования.
5. Назовите типы данных в языке Паскаль.

6. Приведите примеры стандартных функций языка Паскаль.

7. Классификация операторов языка Паскаль. Приведите примеры операторов присваивания.

8. Найдите среди приведённых ниже последовательностей символов те, которые могут быть именами переменных:

а) A;	б) 1B_C;	в) B1_C;	г) hello;
д) ABS(x);	е) a*4;	ж) 1 + 5;	з) Arctan;
и) !!;	к) ?P;	л) A_1Pro;	м) _B1 .

9. Записать по правилам языка Паскаль выражение, значением которого является:

а) площадь квадрата с периметром P;

б) полусумма площади и периметра прямоугольника со сторонами A и B.

10. Запишите на языке Паскаль следующие формулы:

а) $\dfrac{(a+b+c)^2}{a+b}-\dfrac{2\sqrt{ab}}{b-c}$;

б) $\dfrac{(1+x)^3}{|x+y|}-\dfrac{1}{x}+\dfrac{|x-y|}{xy}$;

в) $\dfrac{(x+y)^4+z^3}{|x-y-z|}-\dfrac{\sqrt{x+y}-\sqrt{z}}{(x+y-z)^5}$;

г) $\dfrac{|pq-1|^3}{\left(p^3+q^2\right)^4}-\dfrac{\left(p^2-q^3\right)^5}{\sqrt{1+pq}}$.