

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Национальный технический университет
„Харьковский политехнический институт”

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к практическим занятиям по курсу
“Основы информационных технологий и программирование”
для студентов физико-технических специальностей

Утверждено
редакционно-издательским
советом университета,
протокол №3 от 15.12.2005г.

Харьков НТУ “ХПИ” 2006

Методичні вказівки до практичних занять по курсу “Основы інформаційних технологій та програмування” для студентів фізико-технічних спеціальностей / Укл. М.В. Савченко, В.М. Кухаренко. – Харків: НТУ “ХПИ”, 2006. – 87 с.

Укладачі: канд. фіз.-мат. наук М.В. Савченко

канд. техн. наук В.М. Кухаренко

Рецензент доц., канд. техн. наук О.В. Сіра

Кафедра технічної кріофізики

© Савченко М.В., Кухаренко В.М. 2006 р.

Легко сказать, да не легко доказать.

Пословица [1]

Вступление

Настоящие методические указания предназначены студентам, изучающим основы программирования. **В нем собраны задачи предлагаемые студентам в конце лекции в качестве небольших заданий на 5-15 минут для проверки усвоения текущего материала.** Студенты выполняют эти задания письменно. Максимальный балл за правильно выполненное задание равняется 10. К каждому заданию приведены также некоторые теоретические положения, которые должны помочь студенту при ответе на вопросы. Для каждого задания приведено решение одного контрольного варианта. Наборы вопросов подобраны таким образом, чтобы учесть специфику подготовки студентов технической кафедры, т.е. выработки навыков программной реализации численных алгоритмов. При отборе заданий была использована литература [2-4].

Предполагается, что на практических занятиях студенты проходят тесты и пишут программы (сайт курса на портале НТУ «ХПИ» имеет адрес <http://dl.kpi.kharkov.ua/tkf/tkf3p/default.asp>). При работе используется **рейтинговая система оценки знаний студентов.** Суть этой системы следующая. В настоящее время, для студентов кафедры технической криофизики НТУ «ХПИ» разработаны тесты на каждое из 15 практических занятий. В каждом из этих тестов студент должен ответить на 15 карточек из 40-120 возможных. За каждый правильный ответ в тесте студенту начисляется один балл. В начале семестра студенту предоставляются все задания на составление обязательных программ. Все студенческие программы оцениваются одинаково – по 15 баллов за каждую успешно сданную программу. Таким образом, максимальное количество баллов, которое может набрать студент в течение семестра, равняется 600. Для стимулирования равномерной работы студентов в течение семестра, студент обязан сдать тест (набрать не менее 5 баллов) и отчитаться за написанную программу в течение двух недель с момента начала выполнения соответствующего задания. Если студент нарушает это правило, он получает штрафные -5 баллов (за тесты и программу отдельно).

Студент имеет право проходить любой тест произвольное число раз. Для получения максимального балла за разработанную программу учащийся должен устранить все замечания.

Общее количество баллов, набранных студентом в течение семестра, служит основанием для получения итоговой оценки:

Баллы	Оценка	Баллы	Оценка
600-540	5	479-420	3
539-480	4	419-360	2

1. ЛЕКЦИЯ № 1

ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ. АЛГОРИТМЫ ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ. АЛГОРИТМЫ И АЛГОРИТМИЗАЦИЯ. СОСТАВЛЕНИЕ КОНКРЕТНОГО АЛГОРИТМА. РАБОТА В ИНТЕГРИРОВАННОЙ СРЕДЕ TURBO PASCAL.

1.1. Идентификаторы, числа, строки

Идентификатор – имя любого объекта программы - может включать буквы, цифры и символ подчеркивания. Цифры могут присутствовать в любой позиции, кроме первой. Символ подчеркивания приравнивается к букве. Длина идентификатора может быть любой, но значимыми являются только первые 63 символа, и по этим символам все идентификаторы должны быть уникальными. Прописные и строчные буквы в идентификаторах, числах и служебных словах не различаются.

Не разрешается в языке Паскаль использовать в качестве имен служебные слова и стандартные имена, которыми названы стандартные константы, типы, процедуры, функции и файлы.

Для улучшения наглядности программы в нее могут вставляться пробелы. По крайней мере, один пробел требуется вставить между двумя последовательными именами, числами или служебными и стандартными именами. Пробелы нельзя использовать внутри имен и чисел. Примеры имен языка Паскаль: A b12 r1m SIGMA gamma I80_86

Венгерская нотация основывается на следующих принципах:

- имена переменных и функций должны содержать префикс, описывающий их тип;

- имена переменных и функций записываются полными словами или словосочетаниями или их сокращениями, но так, чтобы по имени можно было понять назначение переменной или действие, выполняемое функцией.

Префиксы записываются малыми буквами, первая буква каждого слова - заглавная, префиксы и слова записываются либо слитно, либо через символ _ (подчеркивание).

Для языка Паскаль могут быть рекомендованы следующие префиксы для скалярных переменных и функций:

by	Byte	e	Extended
sh	Shortint	c	Comp
i	Integer	ch	Char
w	Word	b	Boolean
l	Longint	p	Pointer
r	Real	x,y	координаты символа или точки на экране
si	Single		
d	Double		

Для величин структурированного типа могут быть использованы следующие префиксы:

a	Array	re	Record
s	String	f	File
sz	Stringz	t	Text
se	Set		

Числа в языке Паскаль обычно записываются в десятичной системе счисления. Они могут быть целыми и действительными. Положительный знак числа может быть опущен. Целые числа записываются в форме без десятичной точки, например: 217 -45 8954 +483

В языке Turbo Pascal имеется пять стандартных типов целых чисел, характеристики которых приведены в таблице 1.1.

Таблица. 1.1 – Целые типы данных.

Тип	Диапазон значений	Требуемая память (байт)
-----	-------------------	-------------------------

shortint	-128 .. 127	1
integer	-32768 .. 32767	2
longint	-2147483648 .. 2147483647	4
byte	0 .. 255	1
word	0 .. 65535	2

Для быстрой работы с целыми числами определены процедуры:

inc(x) – увеличивает значение переменной на 1.

inc(x,n) – увеличивает значение переменной на *n*.

dec(x) – уменьшает значение переменной на 1.

dec(x,n) уменьшает значение переменной на *n*.

Результат выполнения функции проверки целой величины на нечетность *odd(x)* имеет значение истина, если аргумент нечетный, и значение ложь, если аргумент четный.

Действительные числа записываются в форме с десятичной точкой или в форме с использованием десятичного порядка, который изображается буквой E: 28.6 0.65 -0.018 4.0 5E12 -1.72E9 73.1E-16. В языке Turbo Pascal имеется пять стандартных типов вещественных чисел, характеристики которых приведены в таблице 1.2.

Таблица. 1.2 – Вещественные типы данных.

Тип	Диапазон значений	Количество цифр мантиссы	Требуемая память (байт)
real	2.9e-39 .. 1.7e+38	11	6
single	1.5e-45 .. 3.4e+38	7	4
double	5.0e-324 .. 1.7e+308	15	8
extended	3.4e-4932 .. 1.1e+4932	19	10
comp	-9.2e+18 .. 9.2e+18	19	8

Паскаль допускает запись целых чисел и фрагментов действительных чисел в форме с порядком в шестнадцатеричной системе счисления: \$7F \$40 \$ABCO

Строки в языке Паскаль - это последовательность символов, записанная между апострофами. Если в строке в качестве содержательного символа

необходимо употребить сам апостроф, то следует записать два апострофа.

Примеры строк:

'ХПИ'

'Кафедра технической криофизики'

'ПРОГРАММА' 'АД''ЮТАНТ'

Строка, состоящая из одного символа, называется константой. Для включения в строку символов, не имеющих физического изображения, используется их ASCII-код с символом # перед ним. Если между апострофами нет ни одного символа, то такая строка называется нулевой или пустой строкой.

1.2. Задание для контроля знаний

1.2.1. Составить по правилам языка Паскаль простые переменные, соответствующие перечисленным ниже объектам ([5,6]).

Пример решения задания 1.2.1.

Математический объект: приведенный матричный элемент неприводимого тензорного оператора

$$\langle n' j' \| A_k \| n j \rangle$$

Идентификатор на языке Паскаль

n1j1_ak_nj

Варианты задания 1.2.1.

- | | |
|---|---|
| 1) $3j$ -символ $\{abc\}$ | 2) $3jm$ -символ Вигнера $\begin{pmatrix} a & b & c \\ \alpha & \beta & \gamma \end{pmatrix}$ |
| 3) Коэффициент Клебша-Гордана $C_{aab\beta}^{c\gamma}$ | 4) $6j$ -символ Вигнера $\begin{Bmatrix} a & b & c \\ d & e & f \end{Bmatrix}$ |
| 5) Коэффициент Рака $W(abcd; ef)$ | 6) $9j$ -символ Вигнера $\begin{Bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{Bmatrix}$ |
| 7) Вектор состояния с угловым моментом j и проекцией m, n | 8) Тензор Леви-Чивита ε_{ikl} |

$|njm\rangle$

- | | |
|--|--|
| 9) Полином Гегенбауэра $C_v^\lambda(x_0)$ | 10) Полином Ганкеля второго рода $H_v^{(2)}(x_0)$ |
| 11) Структурная константа метод ККР A_{LL} | 12) Коэффициента Гаунта $C_{LL'L''}$ |
| 13) Коэффициент Клебша-Гордана $C_{l,m_j-m, \frac{1}{2}, m}^{jm_j}$ | 14) Матричные элементы Гамильтониана H_{ij} |
| 15) Постоянная Планка \hbar | 16) Присоединенные полиномы Лежандра $P_l^n(x_0)$ |
| 17) Радиус s -й МТ сферы R_{MT}^s | 18) Обменно-корреляционный потенциал $V_{xc}(r_0)$ |
| 19) Магнетон Бора μ_B | 20) Сдвиг частот основного состояния Δ_c |
| 21) j -е неприводимое проектное представление группы симметрии волнового вектора k
Γ_{mn}^{kj} | 22) Сферическая гармоника Y_{lm} |
| 23) Комплексная сферическая гармоника Y_{lm}^c | 24) Атомная электронная плотность $\rho^{am}(r_0)$ |
| 25) Объем электронной ячейки Ω_0 | 26) Функция Бесселя 1-го порядка $J_1(x_0)$ |

1.2.2. Запишите правильно числа и строки, перечисленных ниже значений различных величин ([7,8]). Укажите необходимый тип данного для хранения этого данного.

Пример решения задания 1.2.2.

Площадь океанов на Земле км^2 равна

361 000 000

Запись числа на языке Паскаль

361000000 или 361e6

Тип longint или real

Варианты задания 1.2.2.

- | | |
|--|--|
| 1) Удельное сопротивление алюминия в мкОм·м равно 0,028 | 2) Температура плавления алюминия в °C равна 658,3 |
| 3) Температура кипения алюминия в °C равна 2300 | 4) Радиус Солнца в метрах равен $6,9599 \cdot 10^8$ |
| 5) Масса Солнца в килограммах равна $1,989 \cdot 10^{30}$ | 6) Численность населения Украины на 1991 год в тысячах человек была равна 51875 |
| 7) Полтавская битва происходила в 1709 году | 8) Лауреатом Нобелевской премии по физике в 1978 году стал советский академик Капица П.Л. |
| 9) Химический элемент сурьма изображается с помощью символа Sb. | 10) Атомная масса золота равна 196,9665 |
| 11) Продолжительность жизни на Земле в секундах приблизительно равна 10^{18} | 12) Время, за которое лопается мыльный пузырь в секундах равно 10^{-3} |
| 13) Масса электрона в килограммах равна $9,1 \cdot 10^{-31}$ | 14) Масса атмосферы Земли в килограммах равна $5,1 \cdot 10^{18}$ |
| 15) Миля международная морская в километрах равна 1,852 | 16) Приставки для образования кратных и дольных единиц мега и фемто равны соответственно 10^6 и 10^{-15} |
| 17) Число квинтиллион равно 10^{18} | 18) Международное сокращение для единиц давления записывается как Pa |
| 19) Римское представление числа 2005 записывается как MMV | 20) Название космического корабля, на котором летала первая женщина-космонавт |

- | | |
|---|--|
| 21) Дата Православной Пасхи в 2006 году равна 23.04 | 22) Терешкова В.Н., назывался Восток 6
Широта географической координаты Марсианской впадины в Тихом океане равна $11^{\circ}19'c$ |
| 23) Время основания заповедника Аскания Нова 1919 | 24) Международный номерной знак для Канады записывается как CDN |
| 25) Спектральный класс звезды β - Центавра созвездия Центавр записывается как B1 II | 26) 1 лошадиная сила в киловаттах равна 1,3596221 |

2. ЛЕКЦИЯ № 2

ВВЕДЕНИЕ ЯЗЫК ПРОГРАММИРОВАНИЯ TURBO PASCAL АЛФАВИТ ЯЗЫКА. СТРУКТУРА ПРОГРАММЫ. ТИПЫ ДАННЫХ. ВЫРАЖЕНИЯ.

2.1. Арифметические выражения

Выражение – это синтаксическая единица языка, определяющая способ вычисления некоторого выражения. Выражение в языке Паскаль формируется в соответствии с рядом правил из констант, переменных, функций, знаков операций и круглых скобок.

Круглые скобки используются для заключения в них части выражения, значения которой необходимо выполнить в первую очередь. В выражении может быть любое количество круглых скобок, причем количество открывающих круглых скобок должно быть равно количеству закрывающих круглых скобок. Части выражений, заключенные в круглые скобки, должны быть либо непересекающимися, либо вложенными друг в друга.

Вычисление значений выражений выполняется в определенном порядке. Начинается вычисление с определения переменных и констант, входящих в выражение. Дальнейшие действия выполняются в соответствии с их приоритетами (таблица 2.1). В пределах одного и того же приоритета действия не обязательно выполняются слева направо. В целях оптимизации программы компилятор может нарушить этот порядок вычисления.

$$\lg x = \log_{10} x = \frac{\ln x}{\ln 10}; \log_a N = \frac{\log_b N}{\log_b a}$$

$$x^y = e^{y \ln x}$$

Таблица. 2.1 – Приоритеты действий при вычислении выражений.

Группа	Тип действия	Операции или элементы
1	Вычисления в круглых скобках	()
2	Вычисление значений функций	Функции
3	Унарные операции	@,not, унарный +, унарные -
4	Операции типа умножения	*, /, div, mod, and, shl, shr
5	Операции типа сложения	+, -, or, xor
6	Операции отношения	=, <, >, <=, >=, in

В языке Паскаль существует ряд заранее разработанных подпрограмм-функций, которые можно использовать как готовые объекты. Арифметические функции можно использовать только с величинами целого и вещественного типа. Их перечень приведен ниже в таблице 2.2.

Таблица. 2.2 – Арифметические функции.

Функция	Назначение	Тип результата
abs(x)	Абсолютное значение аргумента, $ x $	Совпадает с типом x
arctan(x)	Арктангенс аргумента	Вещественный
	Если $x > 0$, то $\arccos x = \arctg(\sqrt{\frac{1}{x^2} - 1})$	
	Если $0 \leq x < 1$, то	
	$\arcsin x = \arctg(\sqrt{\frac{x^2}{1-x^2}})$	
cos(x)	Косинус аргумента, $\cos x$	Вещественный
exp(x)	Экспоненциальная функция, e^x	Вещественный
frac(x)	Дробная часть числа, $\{x\} = x - x $	Вещественный
int(x)	Целая часть числа x (наибольшее целое число, не превышающее x), $[x]$	Вещественный
ln(x)	Натуральный логарифм, $\ln x = \log_e x$	Вещественный

pi	Значение величины $\pi = 3.1415926535897932385$	Вещественный
round(x)	Округление аргумента до целого значения	Целое
sin(x)	Синус аргумента, $\sin x$	Вещественный
sqr(x)	Квадрат аргумента, x^2	Совпадает с типом x
sqrt(x)	Квадратный корень аргумента, \sqrt{x}	Вещественный
trunc(x)	Отбрасывание дробной части числа	Целое

Арифметические операции применимы только к величинам целых и вещественных типов. Их можно разделить на унарные и бинарные операции. В языке Паскаль отсутствует операция возведения в степень. Арифметические выражения, как и все остальные конструкции языка программирования, вводятся с клавиатуры подряд, один ряд. Бинарные арифметические операции и их знаки приведены в таблице 2.3.

Таблица. 2.3 – Арифметические операции.

Знак	Операция	Тип операндов	Тип результата
+	Сложение	Целые Хотя бы один вещественный	Целый Вещественный
-	Вычитание	Целые Хотя бы один вещественный	Целый Вещественный
*	Умножение	Целые Хотя бы один вещественный	Целый Вещественный
/	Деление	Целые или вещественные	Вещественный
div	Деление целых чисел	Целые	Целый
mod	Остаток от деления целых чисел	Целые	Целый

В операциях деления делитель не должен равняться нулю.

2.2. Задание для контроля знаний

Записать в виде простых арифметических выражений языка Паскаль следующие алгебраические выражения.

Пример решения задания 2.2.

Математическая формула

$$\frac{\sqrt[4]{0,0792(\sin 1 + \cos 1)}}{2,15\sqrt[3]{12,76^2 \operatorname{tg} 4}}$$

Выражение на языке Паскаль

$$\exp(\ln(0.0792*(\sin(1)+\cos(1)))/4)/(2.15*\exp(\ln(\operatorname{sqr}(12.76))/3)*\sin(4)/\cos(4))$$

Варианты задания 2.2.

- 1) а) $0,75\sqrt{0,5} - \frac{1}{2}\sqrt[3]{4}$ б) $100^{\frac{1}{2} \ln 9 - \ln 2} \operatorname{tg}\left(\frac{1}{3}\right)$
- 2) а) $4^{-0,25} - (2\sqrt{2})^{-4/3} \operatorname{tg} 4$ б) $\cos\left(2 \operatorname{arctg} \frac{1}{5} + \operatorname{arctg} \frac{1}{4}\right)$
- 3) а) $86,9^{-\frac{1}{4}} + \left(\frac{1}{2^{-0,3}}\right)^{-\frac{1}{3}}$ б) $49^{1-\lg 2} + 5^{-\lg 4}$
- 4) а) $\frac{8,15\sqrt[3]{14,36} \ln 2}{24,38\sqrt{8,734}(e^2 - e^{-2})}$ б) $\sin\left(\arcsin \frac{1}{2} + \arccos \frac{1}{3}\right)$
- 5) а) $22,5^{-\frac{1}{2}} - 7,5\left(\sqrt[4]{2,87}\right)^2 \cos 1$ б) $\frac{\cos 5}{4 - \sqrt{11}} + \frac{\sin 1}{3 + \sqrt{7}}$
- 6) а) $2\left(\arcsin \frac{5}{13} + \arcsin \frac{12}{13}\right) \ln 3$ б) $2\left(\arcsin \frac{5}{13} + \arcsin \frac{12}{13}\right) \ln 3$
- 7) а) $\frac{12,48\sqrt[3]{5,76} \sin 4}{(1,842)^4 \sqrt[3]{673,8} \cos 8}$ б) $2\left(\arcsin \frac{5}{13} + \arcsin \frac{12}{13}\right) \ln 3$
- 8) а) $\left(0,027^{-\frac{1}{3}} - \left(\frac{1}{6}\right)^{-2,2}\right) \ln 3$ б) $3 \sin 1 + \cos 1$

- 9) а) $\sqrt[5]{\frac{25 + \sqrt{136}}{0,00034}}$ б) $\operatorname{arctg}\left(\cos \frac{\pi}{5} + \cos \frac{2\pi}{5}\right) \ln 5$
- 10) а) $\sqrt{\frac{2,591\sqrt[3]{0,0836}}{1,147(e^2 + e^{-2})}}$ б) $\sqrt[3]{-\lg 0,8 \operatorname{tg} 4}$
- 11) а) $\sqrt[5]{7,002\sqrt[3]{0,1} - 1 + \frac{1}{10}(e^2 + e^{-2})}$ б) $\ln 3\left(\cos \frac{\pi}{5} + \cos \frac{3\pi}{5}\right)$
- 12) а) $\sqrt[10]{10 + \sqrt[10]{10}} \operatorname{tg} 1$ б) $(1 + \sqrt[5]{\lg 20})^{0,2}$
- 13) а) $\sqrt[3]{4,2013\sqrt{0,1 + 2 - \frac{1}{3}(e^2 + e^{-2})}}$ б) $\sin\left(\frac{1}{2} \operatorname{arctg}\left(-\frac{3}{4}\right) \ln 5\right)$
- 14) а) $\frac{4 - 0,0186^2}{\sqrt{0,1} - \sqrt{10}} \operatorname{tg} 2$ б) $\sin((1 + \sqrt[3]{\lg 3})^4)$
- 15) а) $\frac{3,78(e^4 - e^3)}{\sqrt[3]{4} + \sqrt[3]{3}}$ б) $(\ln 3) \sin\left[\frac{1}{2} \arcsin\left(-\frac{2\sqrt{2}}{3}\right)\right]$
- 16) а) $\frac{\log_2 5\sqrt{5} - \sqrt[3]{5} \log_3 5}{1 - 0,1845(\sin 1 + 2 \cos 1)}$ б) $e^{-2} \operatorname{ctg}\left[\frac{1}{2} \arccos\left(-\frac{4}{7}\right)\right]$
- 17) а) $\sqrt{\frac{12,4e + 0,6e^{-1}\sqrt[3]{0,0548}}{0,389(\ln 3 + \sin 1)}}$ б) $\operatorname{tg}\left[5 \operatorname{arctg} \frac{\sqrt{3}}{3} - \frac{1}{4} \arcsin \frac{\sqrt{3}}{2}\right]$
- 18) а) $\frac{1,592^2}{\sqrt[3]{0,382}} \sin 3 + \frac{\sqrt[4]{0,0896}}{0,5348^2} \cos 3$ б) $e^2 \sin\left(3 \operatorname{arctg} \sqrt{3} + 2 \arccos \frac{1}{2}\right)$
- 19) а) $\sqrt[3]{79,836 \ln 3 - \sqrt{156,374} \ln 5}$ б) $(\operatorname{tg} 4) \cos\left[3 \arcsin \frac{\sqrt{3}}{2} + \arccos \frac{1}{2}\right]$
- 20) а) $\left(\frac{1}{3}\right)^{0,2073} \sin 4 - \frac{35}{19} \cos 4$ б) $\lg 2e^{-4\left[\operatorname{arctg}(3+2\sqrt{2}) - \operatorname{arctg} \frac{\sqrt{2}}{2}\right]}$
- 21) а) $0,171^{1,163} \log_2 5 + 2,526 \log_3 7$ б) $(\operatorname{tg} 6)e^{-\left[\arccos \sqrt{\frac{2}{3}} - \arccos \frac{\sqrt{6+1}}{2\sqrt{3}}\right]}$
- 22) а) $\log_6 3,3 - 2(\sqrt[6]{0,6})^{3,3} e^{-2}$

23) а) $2,56^{0,75} \sin 2 + 5,5^{0,33} \sin 3$
 б) $(3e)^{-1} \sin 2,3$

24) а) $0,461^{0,461} \sin 3 - 0,356^{0,356} \cos 3$
 б) $(tg 4) \cos \left(\arcsin \frac{4}{5} + \arcsin \frac{3}{5} + \arcsin \frac{16}{65} \right)$

25) а) $(0,273 \ln 3)^{1,573 \ln 5}$
 б) $\frac{\pi}{3} \ln 2 + \sin \left[\arccos\left(-\frac{1}{7}\right) - \arccos\left(-\frac{13}{14}\right) \right]$

26) а) $\frac{\sqrt[3]{123,4^2 \ln 2}}{1,124 \sqrt[3]{0,024 \cos 1}}$
 б) $\arcsin \left[-\frac{1}{4} (\arctg 2 + \arctg 3) \right]$

<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>		<i>false</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>true</i>		<i>true</i>	<i>true</i>	<i>false</i>

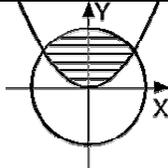
Операции отношения предназначены для сравнения двух величин (величины должны быть сравнимых типов). Результат сравнения имеет логический тип. Операции отношения следующие:

= равно; <= меньше или равно;
 <> не равно; >= больше или равно;
 < меньше; > больше.

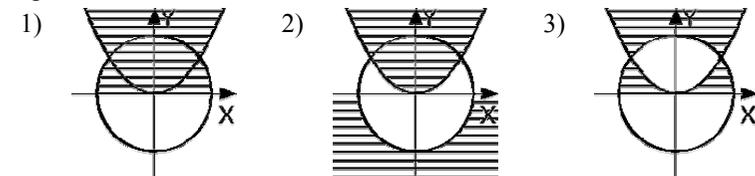
3.2. Задание для контроля знаний

3.2. Записать простое логическое выражение, принимающее значение **true** лишь в случае, когда точка (x,y) находится внутри заштрихованной на рисунке области. На рисунках изображена окружность с единичным радиусом и одна из парабол $y=x^2$, $y=-x^2$, $x=y^2$.

Пример решения задания 3.2.

Исходный рисунок	Логическое выражение на языке Паскаль
	$(x*x+y*y<1) \text{ and } (y>x*x)$

Варианты задания 3.2.



3. ЛЕКЦИЯ № 3

ИСПОЛНЯЕМЫЕ ОПЕРАТОРЫ ПРОСТЫЕ И СТРУКТУРИРОВАННЫЕ
 ОПЕРАТОРЫ. УСЛОВНЫЕ ОПЕРАТОРЫ. ОПЕРАТОРЫ ЦИКЛА.

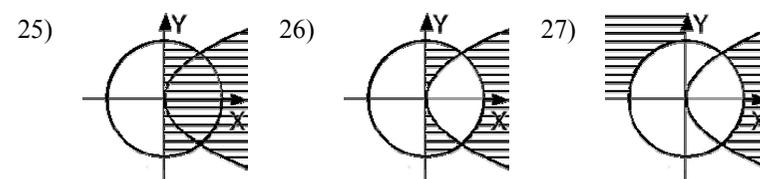
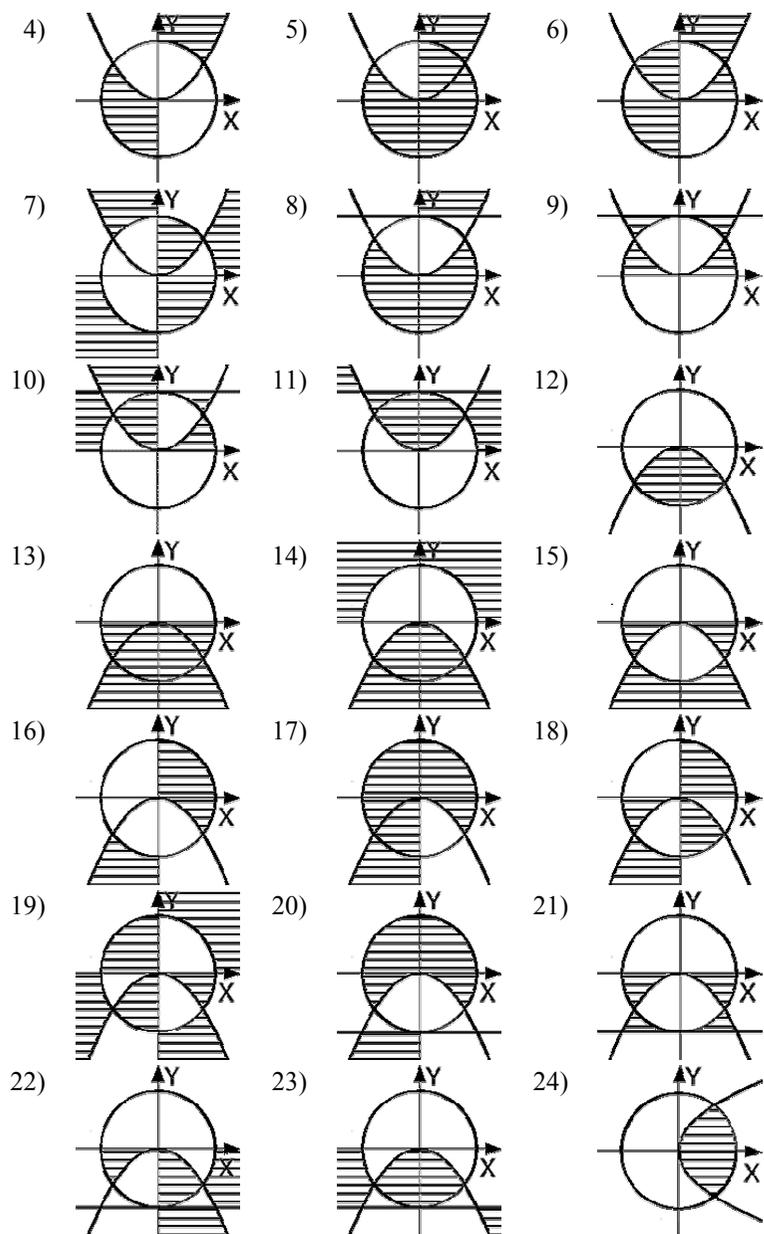
3.1. Логические операции

Стандартный логический тип *boolean* (размер – 1 байт) представляет собой тип данных, любой элемент которого может принимать лишь два значения: *true* и *false*.

Логические операции применяются к величинам логического типа, результат операции – тоже логического типа. Имеется одна унарная логическая операция *not* (отрицание) и три бинарные операции *and* (и), (или) *xor* (исключающее или).

Таблица. 3.1 – Таблица истинности для логических операций.

<i>x</i>	<i>y</i>	<i>not x</i>	<i>x and y</i>	<i>x or y</i>	<i>x xor y</i>
----------	----------	--------------	----------------	---------------	----------------



4. ЛЕКЦИЯ № 4

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ МАССИВЫ. СТРОКИ. ЗАПИСИ.
МНОЖЕСТВА. ФАЙЛОВЫЙ ТИП. СОВМЕСТИМОСТЬ ТИПОВ.

4.1. Простая линейная программа

В программе, написанной на стандарте языка Паскаль, могут следующие разделы:

- заголовок программы;
- раздел объявления меток;
- раздел объявления констант;
- раздел объявления типов;
- раздел объявления переменных;
- раздел объявления процедур и функций;
- тело программы (обязательная часть).

Заголовок программы состоит из зарезервированного слова *program* и имени программы. Завершается заголовок точкой с запятой. Разделы должны обязательно располагаться в указанном порядке.

В языке Turbo Pascal порядок размещения разделов произвольный, можно создавать несколько одинаковых разделов. В любом месте программы можно использовать лишь элементы, которые были определены ранее по тексту программы или являющиеся предопределенными элементами языка. Кроме того, Turbo Pascal существует еще один раздел – раздел объявления используемых модулей.

Тело программы начинается словом *begin*, а заканчивается словом *end* точкой, которая является признаком конца программы.

Пример программы для вычисления суммы двух чисел.

```

program prim;                                {заголовок программы}
var                                           {раздел объявления переменных }

```

<i>x,y,sum: real;</i>	<i>{тело программы}</i>
<i>begin</i>	<i>{вывод запроса на ввод чисел }</i>
<i>write('Введите числа x и y ');</i>	<i>{чтение двух чисел}</i>
<i>readln(x,y);</i>	<i>{определение суммы}</i>
<i>sum:=x+y;</i>	<i>{вывод результата}</i>
<i>writeln('Сумма чисел x и y равна ,sum);</i>	
<i>end.</i>	

Все операторы языка Паскаль можно разбить на две группы: простые и структурированные.

Простыми являются те операторы, которые не содержат в себе других операторов. К ним относятся:

- оператор присваивания;
- оператор обращения с процедуре;
- оператор безусловного перехода goto;
- пустой оператор.

С помощью оператора присваивания переменной или функции присваивается значение выражения. Для этого используется знак присваивания :=, слева от которого записывается имя переменной или функции, которой присваивается значение, а справа – выражение, значение которого вычисляется перед присваиванием.

Пустой оператор не выполняет никакого действия и никак не отображается в программе. Он может потребоваться для осуществления безусловного перехода.

Для ввода исходных данных используются операторы процедур ввода:

```
read(a1,a2,...ak);
readLn(a1,a2,...ak);
readLn;
```

Первый из них реализует чтение *k* значений исходных данных и присваивание этих значений переменным *a1, a2, ..., ak*. Второй оператор реализует чтение *k* значений исходных данных, пропуск остальных значений до начала следующей строки, присваивание считанных значений переменным *a1, a2, ..., ak*. Третий оператор реализует пропуск строки в исходных данных.

При вводе исходных данных происходит преобразование из внешней формы представления во внутреннюю, определяемую типом переменных. Переменные, образующие список ввода, могут принадлежать либо к целому, либо к действительному, либо к символьному типу. Чтение исходных данных логического типа в языке Паскаль недопустимо.

Операторы ввода при чтении значений переменных целого и действительного типа пропускает пробелы, предшествующие числу. В то же время эти операторы не пропускают пробелов, предшествующих значениям символьных переменных, так как пробелы являются равноправными символами строк. Пример записи операторов ввода:

```
var rv, rs: real;
    iw, ij: integer;
    chc, chd: char;
.....
read(rv, rs, iw, ij);
read(chc, chd);
```

Значения исходных данных могут отделяться друг от друга пробелами и нажатием клавиш табуляции и *enter*.

Для вывода результатов работы программы на экран используются операторы:

```
write(a1,a2,...ak);
writeln(a1,a2,...ak);
writeln;
```

Первый из этих операторов реализует вывод значений переменных *a1, a2, ..., ak* в строку экрана. Второй оператор реализует вывод значений переменных *a1, a2, ..., ak* и переход к началу следующей строки. Третий оператор реализует пропуск строки и переход к началу следующей строки.

Переменные, составляющие список вывода, могут относиться к целому, действительному, символьному или булевскому типам. В качестве элемента списка вывода кроме имен переменных могут использоваться выражения и строки.

Вывод каждого значения в строку экрана происходит в соответствии с шириной поля вывода, определяемой конкретной реализацией языка.

Форма представления значений в поле вывода соответствует типу переменных и выражений: величины целого типа выводятся как целые

десятичные числа, действительного типа - как действительные десятичные числа с десятичным порядком, символьного типа и строки - в виде символов, логического типа - в виде логических констант *true* и *false*.

Оператор вывода позволяет задать ширину поля вывода для каждого элемента списка вывода. В этом случае элемент списка вывода имеет вид *a:k*, где *a* - выражение или строка, *k* - выражение либо константа целого типа.

Если выводимое значение занимает в поле вывода меньше позиций, чем *k*, то перед этим значением располагаются пробелы. Если выводимое значение не помещается в ширину поля *k*, то для этого значения будет отведено необходимое количество позиций. Для величин действительного типа элемент списка вывода может иметь вид *a:k:m*, где *a* - переменная или выражение действительного типа, *k* - ширина поля вывода, *m* - число цифр дробной части выводимого значения. *k* и *m* - выражения или константы целого типа. В этом случае действительные значения выводятся в форме десятичного числа с фиксированной точкой.

Пример записи операторов вывода:

```
var ra, rb: real;
    ip, iq: integer;
    br, bs: boolean;
    cht, chv, chu, chw: char;
```

```
.....
writeln(ra, rb:10:2);
writeln(ip, iq:8);
writeln(br, bs:8);
writeln(cht, chv, chu, chw);
```

4.2. Задание для контроля знаний

Составить программу на языке Паскаль для вычисления по заданным формулам.

Пример решения задания 4.2.

Исходные математические формулы

$$a = u \frac{x+y}{2} - 3 \sqrt[3]{\frac{x-1}{|y|+1}}, b = \sin(2 \arccos v),$$

где $x=12,650$, $y=-2,255$, $u=3,205$, $v=0,880$.

Программа на языке Паскаль

```
program prim;
var
a,b,x,y,u,v: real;
begin
x:=12.650;
y:=-2.255;
u:=3.205;
v:=0.88;
a:=exp((x+y)*ln(u)/2)-exp(ln((x-1)/(abs(y)+1))/3);
b:=sin(2*arctan(sqrt(1/v/v-1)));
write(a:12:5,b:12:5);
end.
```

Варианты задания 4.2.

$$1) \quad a = 2^{-x} \sqrt{x + \sqrt[4]{|y|}}$$

$$b = \sqrt[3]{e^{x - \frac{1}{\sin z}}}$$

$x=3,981 \quad y=-1,625 \quad z=0.512$

$$2) \quad a = y^{\sqrt[3]{|x|}} + ch^3(y-3)$$

$$b = \frac{y(\arctgz - \frac{\pi}{6})}{|x| + \frac{1}{y^2 + 1}}$$

$x=-6,251 \quad y=0,827 \quad z=25,001$

$$3) \quad a = 2^{y^x} + (3^x)^y$$

$$b = \frac{|x-y| \left(1 + \frac{\sin^2 z}{x+y} \right)}{e^{|x-y|} + \frac{x}{2}}$$

$x=3,251 \quad y=0,325 \quad z=0,466$

$$4) \quad a = \frac{\sqrt{|x-1|} - \sqrt[3]{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}$$

$$b = x(\arctgz + e^{-(x+3)})$$

$x=-6,222 \quad y=3,325 \quad z=5,541$

$$5) \quad a = \sqrt[4]{y} + \sqrt[3]{x} - 1$$

$$b = |x-y|(\sin^2 z + \operatorname{tg} z)$$

$x=17,421 \quad y=10,365 \quad z=0,828$

$$6) \quad a = \frac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \frac{x + \frac{y}{2}}{2|x+y|}$$

$$b = (x+1) \frac{1}{\sin z}$$

$x=1,625 \quad y=-15,400 \quad z=0,252$

7)
$$a = \frac{x^{y+1} + e^{y-1}}{1 + x|y - tz|}$$

$$b = 1 + |y - x| + \frac{|y - x|^2}{2} + \frac{|y - x|^3}{3}$$

$$x = 2,444 \quad y = 0,869 \quad z = -0,166$$

9)
$$a = (1 + y) \frac{x + \frac{y}{x^2 + 4}}{y^{x-2} + \frac{1}{x^2 + 4}}$$

$$b = \frac{1 + ch(y - 2)}{\frac{x}{2} + \sin^2 z}$$

$$x = 3,258 \quad y = 4,005 \quad z = -0,666$$

11)
$$a = \lg(\sqrt{e^{x-y}} + x^{|y|} + z)$$

$$b = x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

$$x = 1,542 \quad y = -3,261 \quad z = 80,005$$

13)
$$a = \frac{1 + sh^2(x + y)}{\left| x - \frac{2y}{1 + x^2 y^2} \right|} x^{|y|}$$

$$b = \cos^2\left(\arctg \frac{1}{z}\right)$$

$$x = 3,741 \quad y = -0,825 \quad z = 0,160$$

15)
$$a = |\cos x + \cos y|^{1 + 2 \sin^2 y}$$

$$b = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!}$$

$$x = 0,400 \quad y = -0,875 \quad z = -0,475$$

8)
$$a = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$$

$$b = x(\sin \arctgz + \cos^2 y)$$

$$x = 0,335 \quad y = 0,025 \quad z = 32,005$$

10)
$$a = y + \frac{x}{y + \frac{x^2}{y + \frac{x^3}{y}}}$$

$$b = (1 + tg^2 \frac{z}{2})^{\sqrt{|y|+6}}$$

$$x = 0,100 \quad y = -8,870 \quad z = 0,765$$

12)
$$a = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{\frac{1}{2} + \sin^2 y}$$

$$b = 1 + \frac{z^2}{3 + \frac{z^2}{5}}$$

$$x = 1,426 \quad y = -1,220 \quad z = 3,500$$

14)
$$a = \frac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2}$$

$$b = e^{|x-y|} (tg^2 z + 1)^z$$

$$x = -4,500 \quad y = 0,750 \quad z = 0,845$$

16)
$$a = \ln\left(y^{-\sqrt{|x|}}\right) \left(x - \frac{y}{2}\right)$$

$$b = \sin^2 \arctgz$$

$$x = -15,246 \quad y = 4,642 \quad z = 20,001$$

17)
$$a = \sqrt{10(\sqrt[3]{x} + x^{y+2})}$$

$$b = \arcsin^2 z + |x + y|$$

$$x = 16,55 \quad y = -2,75 \quad z = 0,15$$

19)
$$a = e^{|x-y|} + |x - y|^{x+y}$$

$$b = \arctgx + \arctgz$$

$$x = -2,235 \quad y = -0,823 \quad z = 15,221$$

21)
$$a = \frac{x + \frac{y}{5 + \sqrt{x}}}{|y - x| + \sqrt{x}}$$

$$b = e^{u-1} + \arcsin v$$

$$x = 47,8 \quad y = -5,5 \quad u = -2,3 \quad v = -0,8$$

23)
$$a = \sqrt[3]{x + \sqrt[4]{|y|}}$$

$$b = \sqrt{|y|} e^{-(y + \frac{u}{2})}$$

$$x = 37,15 \quad y = -12,55 \quad u = 20,12$$

25)
$$a = (2 + y^2) \frac{x + \frac{y}{2}}{y^2 + \frac{1}{1 + y^2}}$$

$$b = \sqrt{\sin^2 \arctgu + |\cos v|}$$

$$x = 0,22 \quad y = -6,72 \quad u = 10,05 \quad v = 0,35$$

18)
$$a = 5 \arctgx - \frac{1}{4} \arctgy$$

$$b = \frac{x + 3|x - y| + x^2}{|x - y|^2 + x^2}$$

$$x = -17,22 \quad y = 6,33 \quad z = 3,25$$

20)
$$a = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right|$$

$$b = (y - x) \frac{y - z}{1 + (y - x)^2}$$

$$x = 1,825 \quad y = 18,225 \quad z = -3,298$$

22)
$$a = y^x + \sqrt[3]{|x| + |y|}$$

$$b = u + \frac{v^3}{u + \frac{v^3}{u + v^3}}$$

$$x = -0,85 \quad y = 1,25 \quad u = -0,22 \quad v = 0,01$$

24)
$$a = \frac{1}{2} \left(x^{|y-x|} + y^{\frac{x+y}{2}} \right)$$

$$b = \lg(\sqrt[3]{u} + \sqrt{v} + 2)$$

$$x = 3,255 \quad y = 2,981 \quad u = 125,331 \quad v = 33,075$$

26)
$$a = u^{\frac{x+y}{2}} - \sqrt[3]{\frac{x-1}{|y|+1}}$$

$$b = \sin(2 \arccos v)$$

$$x = 12,650 \quad y = -2,255 \quad u = 3,205 \quad v = 0,880$$

5. ЛЕКЦИЯ № 5
ПРОЦЕДУРЫ И ФУНКЦИИ ПРОЦЕДУРА. ФУНКЦИЯ. ФОРМАЛЬНЫЕ И

5.1. Структурированные операторы

Структурированными являются такие операторы, которые состоят из других операторов. К ним относятся:

- составной оператор (блок);
- условный оператор if;
- условный оператор case;
- оператор цикла repeat;
- оператор цикла while;
- оператор цикла for;
- оператор работы над записями with.

Составной оператор представляет собой совокупность последовательно выполняемых операторов, заключенных в операторные скобки *begin* и *end*:

```
begin
<оператор 1>;
<оператор 1>;
...
<оператор n>
end;
```

Он требуется в тех случаях, когда в соответствии с правилами построения конструкций языка можно использовать один оператор, а выполнить необходимо несколько действий.

Условный оператор if реализует алгоритмическую конструкцию развилка и изменяет порядок выполнения операторов в зависимости от истинности или ложности некоторого условия. Существует два варианта оператора

```
If S then A
Else B;      {полная развилка}
и
If S then A  {укороченная развилка}
```

В этих операторах:

S – некоторое логическое выражение, истинность которого проверяется;

A – оператор, который выполняется, если выражение *S* истинно;

B – оператор, который выполняется, если выражение *S* ложно.

Так условный оператор *if* является единым предложением, ни перед *then*, ни перед *else* точку с запятой ставить нельзя (смотри рисунок 5.1).

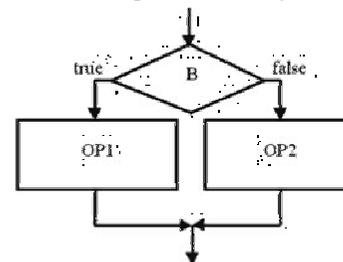


Рис.5.1– Структурные схемы условного оператора if B then OP1 else OP2;

5.2. Задание для контроля знаний

Составить программу на языке Паскаль для вычисления по заданным формулам с использованием операторов ветвления.

Пример решения задания 5.2.

Исходные математические формулы

$$\mu = \begin{cases} \frac{|\alpha| - |\beta|}{\alpha^2 + \beta^2} & \text{при } \alpha\beta > \frac{1}{2} \\ |\alpha + \beta| & \text{при } \alpha\beta \leq \frac{1}{2} \end{cases}$$

Программа на языке Паскаль

```
program prim;
var
mu,alpha,betta,ab: real;
begin
writeln('Введите коэффициенты alpha и betta');
read(alpha,betta);
ab:=alpha*betta;
if ab > 0.5 then
mu:=exp(-(abs(alpha)-abs(betta))/(alpha*alpha+betta*betta))
else
```

```

mu:=abs(alpha+beta);
writeln('mu= ',mu:12:5,' alpha*beta= ',ab:12:5);
end.

```

Варианты задания 5.2.

- 1) $k = \begin{cases} \sqrt{15a^2 + 21b^2} \text{ npu } a > b \\ \sqrt{15b^2 + 21a^2} \text{ npu } a \leq b \end{cases}$
- 2) $l_z = \begin{cases} \ln(|2l_x - 3e^2l_y|) \text{ npu } |l_x| < 5|l_y| \\ \ln(|2l_x e^2 - 3l_y|) \text{ npu } |l_x| \geq 5|l_y| \end{cases}$
- 3) $p = \begin{cases} \sin(5k + 3m \ln 3) \text{ npu } |k| > |m| \\ \cos(5k + 3m \ln 3) \text{ npu } |k| \leq |m| \end{cases}$
- 4) $r = \begin{cases} \sqrt{|2k_1 - 7k_2|} \text{ npu } \min(k_1, k_2) < 1 \\ \sqrt{|2k_1 + 7k_2|} \text{ npu } \min(k_1, k_2) \geq 1 \end{cases}$
- 5) $s = \begin{cases} \frac{4r + 3m}{r^2 + m^2} \text{ npu } |r| > |m| + \frac{1}{2} \\ |r - m| \text{ npu } |r| \leq |m| + \frac{1}{2} \end{cases}$
- 6) $\zeta = \begin{cases} \sqrt{3\xi^2 + 4\eta^2} \text{ npu } |\xi| \leq 2|\eta| \\ \sqrt{3\xi^2 - 4\eta^2} \text{ npu } |\xi| > 2|\eta| \end{cases}$
- 7) $n = \begin{cases} \frac{s - 2t}{2s^2 + 5t^2} \text{ npu } st < 0 \\ \sqrt{st} \text{ npu } st \geq 0 \end{cases}$
- 8) $l = \begin{cases} th(c - 2k) \text{ npu } |c + k| > 2 \\ \ln(|c - 2k|) \text{ npu } |c + k| \leq 2 \end{cases}$
- 9) $m = \begin{cases} \frac{e^{-u_i} + e^{-v_i}}{2|u_i| + 3|v_i|} \text{ npu } 2|u_i| < v_i \\ u_i + v_i \text{ npu } 2|u_i| \geq v_i \end{cases}$
- 10) $u = \begin{cases} \frac{3l_1 - 5l_2}{l_1^2 + l_2^2} \text{ npu } |l_1| < 1 + |l_2| \\ \frac{3l_1 + 5l_2}{l_1^2 - l_2^2} \text{ npu } |l_1| \geq 1 + |l_2| \end{cases}$
- 11) $s = \begin{cases} \arctg(5m_i^2 + 7n_i^2) \\ \text{npu } m_i^2 + n_i^2 > 0,1 \\ \arcsin(5m_i^2 + 7n_i^2) \\ \text{npu } m_i^2 + n_i^2 \leq 0,1 \end{cases}$
- 12) $n_3 = \begin{cases} \sin(\pi n_1 + e^{n_2}) \text{ npu } n_1 + n_2 < 5 \\ \sin(\pi n_1 + n_2) \text{ npu } n_1 + n_2 \geq 5 \end{cases}$
- 13) $k = \begin{cases} \sqrt{|3m - 5r|} \text{ npu } m < 2r \\ \sqrt{|3m + 5r|} \text{ npu } m \geq 2r \end{cases}$
- 14) $l = \begin{cases} \sqrt{|d + c|} \text{ npu } d^2 + c^2 > 10 \\ d + c \text{ npu } d^2 + c^2 \leq 10 \end{cases}$

- 15) $m_t = \begin{cases} \frac{m_t - 2m_s}{m_r^2 + 2m_s^2} \text{ npu } |m_r - 2m_s| \leq 1 \\ \frac{2}{m_r - 2m_s} \text{ npu } |m_r - 2m_s| > 1 \end{cases}$
- 16) $s = \begin{cases} \sqrt{n_1 n_2} \text{ npu } n_1 n_2 < -0,1 \\ \sqrt{|n_1 + n_2|} \text{ npu } n_1 n_2 \geq -0,1 \end{cases}$
- 17) $l = \begin{cases} \frac{3u_n + v_n}{u_n^2 + v_n^2} \text{ npu } |u_n| < |v_n| \\ u_n v_n \text{ npu } |u_n| \geq |v_n| \end{cases}$
- 18) $k = \begin{cases} \ln(|p| + 5|r|) \text{ npu } p^2 + r^2 > 1 \\ p - |r| \text{ npu } p^2 + r^2 \leq 1 \end{cases}$
- 19) $k = \begin{cases} \sqrt{|se^2 - ne^{-2}|} \text{ npu } s \leq |n| \\ \sqrt{s - n} \text{ npu } s \leq |n| \end{cases}$
- 20) $m = \begin{cases} \sqrt{3|st|} \text{ npu } s \leq t \\ s + t \text{ npu } s > t \end{cases}$
- 21) $u = \begin{cases} \ln(|p| + |n|) \text{ npu } p \leq n + 1 \\ \ln(|p - n|) \text{ npu } p \leq n + 1 \end{cases}$
- 22) $i_3 = \begin{cases} \sqrt{|2i_1 e^3 - 3i_2 e^2|} \text{ npu } i_1 i_2 > 5 \\ \sqrt{|2i_1 + 3i_2|} \text{ npu } i_1 i_2 \leq 5 \end{cases}$
- 23) $m_k = \begin{cases} \frac{m_i - m_j}{3m_i^2 + 4m_j^2} \text{ npu } |m_i| + |m_j| > 1 \\ m_i^2 - m_j^2 \text{ npu } |m_i| + |m_j| \leq 1 \end{cases}$
- 24) $n = \begin{cases} \sqrt{|xe + te^{-1}|} \text{ npu } x < 10t \\ \sqrt{|x + t|} \text{ npu } x \geq 10t \end{cases}$
- 25) $l = \begin{cases} \frac{7k - 5p}{2k^2 + 3p^2} \text{ npu } k > |p| \\ |k - p| \text{ npu } k \leq |p| \end{cases}$
- 26) $s = \begin{cases} e^{-|m+r|} \text{ npu } r > -2m \\ mr \text{ npu } r \leq -2m \end{cases}$

6. ЛЕКЦИЯ № 6

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ УКАЗАТЕЛИ. СПИСКИ. СТЕКИ.
ОЧЕРЕДИ. ДЕРЕВЬЯ. РАБОТА С ПАМЯТЬЮ.

6.1. Операторы цикла

Оператор цикла *repeat* организует выполнение цикла, состоящего из любого числа операторов, с неизвестным заранее числом повторений. Тело цикла выполняется хотя бы один раз. Выход из цикла осуществляется при

истинности некоторого логического выражения. Структура этого оператора следующая:

```
repeat
<оператор 1>;
<оператор 1>;
...
<оператор n>
until S;
```

В этой структуре операторы образуют тело цикла, а S – логическое выражение, истинность которого проверяется в конце каждой итерации (смотри рисунок 6.1).

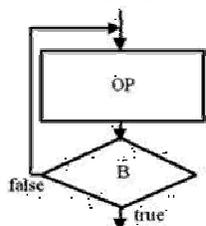


Рис.6.1– Структурная схема, соответствующая оператору repeat OP until B;

Оператор цикла *while* организует выполнение одного оператора неизвестное заранее число раз. Выход из цикла осуществляется, если некоторое логическое выражение окажется ложным. Так как истинность логического выражения проверяется в начале каждой итерации, тело цикла может не выполняться ни разу. Структура оператора имеет вид:

```
while S do <оператор>;
```

В этой структуре S – логическое выражение, истинность которого проверяется в начале каждой итерации (смотри рисунок 6.2).

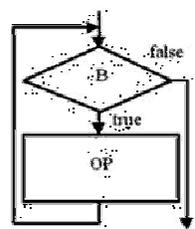


Рис.6.2– Структурная схема, соответствующая оператору while B do OP;

Оператор цикла *for* организует выполнение одного оператора заранее известное число раз. Существует два варианта оператора:

```
for param:=start to finish do <оператор>; (шаг цикла succ(param))
for param:=start downto finish do <оператор>; (шаг цикла pred(param))
```

В этих операторах:

param – параметр цикла, являющийся переменной порядкового типа; start (finish) – выражение, определяющее начальное (конечное) значение параметра цикла.

Вначале работы этого цикла вычисляется, и запоминаются начальное и конечное значения параметров цикла. Далее параметру цикла присваивается начальное значение. Затем значение параметра цикла сравнивается со значением *finish*. Далее, пока параметр цикла меньше или равен конечному значению (первый вариант оператора) или больше или равен конечному значению (во втором варианте), выполняется очередная итерация; в противном случае происходит выход из цикла. После выхода из цикла параметр цикла становится неопределенным (смотри рисунок 6.3).

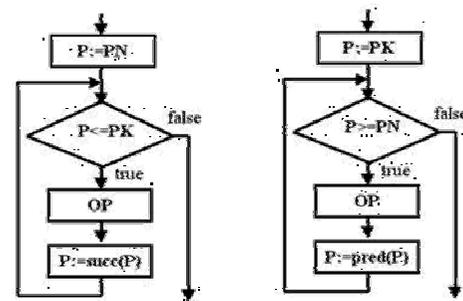


Рис.6.3– Структурные схемы, соответствующие операторам цикла for P:=PN to PK do OP;

и
for P:=PK downto PN do OP;

Пример. Оператор цикла *for C:='A' to 'Z' do OP*;

где C - переменная символьного типа, обеспечит выполнение тела цикла 26 раз, а управляющая переменная C будет принимать значения всех букв латинского алфавита.

6.2. Задание для контроля знаний

Составить программу на языке Паскаль для проверки разложений соответствующих функций.

Пример решения задания 6.2.

Исходные математические формулы

$$\frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots$$

Программа на языке Паскаль

<pre> program prim; var x,xi,f,sum: double; i,n: integer; begin x:=0.25; n:=20; f:=sin(x)/x; sum:=1.0; </pre>	<pre> xi:=1.0; for i:=1 to n do begin xi:=-xi*x*(i+1)/(i+2); sum:=sum+xi end; writeln(sum:15:8,f:15:8,abs(sum-f):15:8); end. </pre>
---	---

Варианты задания 6.2.

$$1) e^{-x^2} = 1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \frac{x^6}{3!} + \dots$$

$$2) \ln(x + \sqrt{x^2 + 1}) = x - \frac{1}{2} \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \frac{x^5}{5} - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \frac{x^7}{7} + \dots$$

$$3) \operatorname{arctg}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

$$4) \operatorname{arcsin}(x) = x + \frac{1}{2} \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \frac{x^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \frac{x^7}{7} + \dots$$

$$5) \frac{1}{\sqrt{1-x^2}} = 1 + \frac{1}{2} x^2 + \frac{1 \cdot 3}{2 \cdot 4} x^4 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} x^6 + \dots$$

$$6) \frac{1}{\sqrt{1+x}} = 1 - \frac{1}{2} x + \frac{1 \cdot 3}{2 \cdot 4} x^2 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} x^3 + \dots$$

$$7) \sqrt{1+x} = 1 + \frac{1}{2} x - \frac{1}{2 \cdot 4} x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} x^3 - \dots$$

$$8) \frac{1}{(1+x)^3} = 1 - \frac{2 \cdot 3}{2} x + \frac{3 \cdot 4}{2} x^2 - \frac{4 \cdot 5}{2} x^3 + \dots$$

$$9) \frac{1}{(1+x)^2} = 1 - 2x + 3x^2 - 4x^3 + \dots$$

$$10) \frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 + \dots$$

$$11) \ln \frac{1+x}{1-x} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots\right)$$

$$12) \ln(1-x) = -\frac{x}{1} - \frac{x^2}{2} - \frac{x^3}{3} - \dots$$

$$13) \ln(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$$

$$14) \operatorname{ch}(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots$$

$$15) \operatorname{sh}(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots$$

$$16) \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

$$17) \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

$$18) e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \dots$$

7. ЛЕКЦИЯ № 7

ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА СТАНДАРТНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ ВСЕХ ТИПОВ. РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ. ТИПИЗИРОВАННЫЕ И НЕТИПИЗИРОВАННЫЕ ФАЙЛЫ.

7.1. Массивы

Массивы представляют собой ограниченную упорядоченную совокупность однотипных величин. Каждая отдельная величина называется компонентом массива. Тип компонент может быть любым, принятым в языке Паскаль, кроме файлового типа. Тип компонент называется базовым типом.

Вся совокупность компонент определяется одним именем. Для обозначения отдельных компонент используется конструкция, называемая переменной с индексом или с индексами: $A[5]$, $S[k+1]$, $B[3,5]$.

В качестве индекса может быть использовано выражение. Тип индексов может быть только интервальным или перечисляемым. Действительный и целый типы недопустимы. Индексы интервального типа, для которого базовым является целый тип, могут принимать отрицательные, нулевое и положительные значения.

В операторной части программы один массив может быть присвоен другому, если их типы идентичны, например: $R1:=Z$;

Для ввода или вывода массива в список ввода или вывода помещается переменная с индексом, а операторы ввода или вывода выполняются в цикле.

Первый индекс определяет номер строки, второй - номер столбца. Двумерные массивы хранятся в памяти компьютера по строкам.

Инициализация массивов (присвоение начальных значений всем компонентам массивов) осуществляется двумя способами.

Первый способ – с использованием типизированных констант, например:

```
type Dim10= Array[1..10] of Real;
const
  raM10: Dim10 = ( 0, 2.1, 4, 5.65, 6.1, 6.7, 7.2, 8, 8.7, 9.3 );
```

При инициализации двумерных массивов значения компонент каждого из входящих в него одномерных массивов записывается в скобках:

```
type Dim3x2= Array[1..3,1..2] of Integer;
const
  iaM3x2: Dim3x2= ( (1, 2) (3, 4) (5, 6) );
```

Второй способ инициализации - использование разновидности процедуры *FillChar*: *FillChar*(var V; NBytes: Word; B: Byte);

Эта процедура заполняет участок памяти однобайтовым значением.

Например, для обнуления массива $A[1..10]$ of Real можно записать:

FillChar(A, 40, 0); или *FillChar*(A, SizeOf(A), 0);

7.2. Задание для контроля знаний

Пример решения задания 7.2.

Определить количество нулевых элементов в целочисленном массиве.

Программа на языке Паскаль

<pre>program prim; const l=5; var a : array[1..8] of integer; i,k: integer; begin write('Вводите элементы массива: '); k:=0; for i:=1 to l do</pre>	<pre>begin read(a[i]); if a[i]=0 then k:=k+1 end; writeln('Количество нулевых элементов в массиве: ',k) end.</pre>
---	--

Варианты задания 7.2.

- 1) В массиве 100 случайных реальных чисел от 0 до 1 найти минимум суммы трех элементов.
- 2) Слить два массива А и В по 100 элементов в массив С из 200 элементов так, чтобы элементы массива А имели в С нечетные номера.
- 3) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 1 до 9. Определить по А и вывести массив В 10x10 так, чтобы элементы в последней строке определялись как суммы элементов по соответствующим столбцам.
- 4) Сгенерировать и вывести на экран массив А 10x9 случайных целых чисел в диапазоне от 1 до 9. Определить и вывести производный от А массив В 10x10 так, чтобы элементы в последней строке определялись как суммы элементов по соответствующим столбцам.
- 5) В массиве 100 случайных реальных чисел от 0 до 1 найти максимум суммы трех элементов.
- 6) Слить два массива А и В по 100 элементов в массив С из 200 элементов так, чтобы элементы массива А имели номера от 51 до 150.
- 7) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 1 до 10. Определить по А и вывести массив В 10x10 так, чтобы элементы в последнем столбце определялись как произведение элементов по соответствующим строкам.
- 8) Сгенерировать и вывести на экран массив А 9x10 случайных целых чисел в диапазоне от 2 до 5. Определить и вывести производный от А

массив В 10x10 так, чтобы элементы в последнем столбце определялись как произведение элементов по соответствующим строкам.

- 9) В массиве 100 случайных реальных чисел от 0 до 1 найти минимум суммы двух элементов.
- 10) Слить два массива А и В по 100 элементов в массив С из 200 элементов так, чтобы элементы А и В чередовались по 10 штук.
- 11) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 1 до 3. Определить по А и вывести массив В 10x10 так, чтобы элементы главной диагонали определялись как суммы по соответствующим строкам.
- 12) Сгенерировать и вывести на экран массив А 9x10 случайных целых чисел в диапазоне от 1 до 4. Определить и вывести производный от А массив В 10x10 так, чтобы элементы главной диагонали определялись как суммы по соответствующим строкам.
- 13) В массиве 100 случайных реальных чисел от 0 до 1 найти максимум суммы двух элементов.
- 14) Слить два массива А и В по 100 элементов в массив С из 200 элементов так, чтобы вначале шли элементы меньше среднего значения по всему массиву С.
- 15) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 2 до 4. Определить по А и вывести массив В 10x10 так, чтобы элементы главной диагонали определялись как произведение по соответствующим столбцам.
- 16) Сгенерировать и вывести на экран массив А 9x10 случайных целых чисел в диапазоне от 1 до 10. Определить и вывести производный от А массив В 10x10 так, чтобы элементы главной диагонали определялись как произведение по соответствующим столбцам.
- 17) В массиве 200 случайных реальных чисел от 0 до 1 найти максимум, не превышающий по значению 0.5.
- 18) Слить два массива А и В по 50 элементов в массив С из 100 элементов так, чтобы элементы массива А имели в С четные номера.
- 19) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 2 до 5. Определить по А и вывести

массив В 10x10 так, чтобы элементы главной диагонали определялись как произведение по соответствующим столбцу и строке.

- 20) Сгенерировать и вывести на экран массив А 9x10 случайных целых чисел в диапазоне от 3 до 7. Определить и вывести производный от А массив В 10x10 так, чтобы элементы главной диагонали определялись как произведение по соответствующим столбцу и строке.
- 21) В массиве 200 случайных реальных чисел от 0 до 2 найти минимум, по значению не меньший единицы.
- 22) Слить два массива А и В по 50 элементов в массив С из 100 элементов так, чтобы элементы массива А имели номера от 21 до 70.
- 23) Сгенерировать и вывести на экран массив А 10x10 случайных реальных чисел в диапазоне от 1 до 6. Определить по А и вывести массив В 10x10 так, чтобы элементы в последней строке определялись как суммы элементов по соответствующим столбцам.
- 24) Сгенерировать и вывести на экран массив А 10x9 случайных целых чисел в диапазоне от 1 до 9. Определить и вывести производный от А массив В 10x10 так, чтобы элементы в последней строке определялись как суммы элементов по соответствующим столбцам.

8. ЛЕКЦИЯ № 8

**МОДУЛИ ЗАГОЛОВК МОДУЛЯ. ИНТЕРФЕЙС МОДУЛЯ.
ИСПОЛНИТЕЛЬНАЯ ЧАСТЬ МОДУЛЯ. СЕКЦИЯ ИНИЦИАЛИЗАЦИИ.
ИСПОЛЬЗОВАНИЕ МОДУЛЯ В ОСНОВНОЙ ПРОГРАММЕ. СТАНДАРТНЫЕ
МОДУЛИ.**

8.1. Подпрограммы

Подпрограмма – это повторяющаяся группа операторов, оформленная в виде самостоятельной программной единицы. Подпрограммы предназначены для решения отдельных подзадач основной задачи.

Правило предшествования описания Подпрограмма должна быть описана до того, как она будет использована в программе или другой подпрограмме.

В языке Паскаль имеется две разновидности подпрограмм – процедуры и функции.

Программы на языке Паскаль состоят из заголовка программы, раздела описаний и тела программы. Раздел описаний может включать следующие подразделы: объявление меток, констант, типов, переменных, процедур и функций. Последовательность подразделов в структуре программы произвольная, но обязательно должно выполняться правило предшествования описания.

Все параметры (константы, переменные, типы) в программе делятся на глобальные и локальные.

Глобальные параметры – это параметры из раздела описаний основной части программы. Они доступны как в основной программе, так и во всех подпрограммах.

Локальные параметры (константы, переменные, типы) – из раздела описаний процедур и функций. Они доступны только внутри конкретной подпрограммы.

Подпрограмма может непосредственно использовать любые глобальные параметры за исключением тех, которые имеют те же имена, что и ее локальные параметры.

Локальные параметры существуют только в течение времени работы процедуры, определяются (создаются) при её вызове и исчезают после завершения работы.

Обмен информацией между основной программой и подпрограммой может осуществляться только с помощью глобальных параметров.

Подпрограмма может использовать глобальные параметры двояким образом: непосредственно обращаясь к глобальному параметру по его имени или используя механизм формальных параметров.

При описании подпрограмм формальные параметры указываются в виде списка имен с типом через точку с запятой и помещаются в круглые скобки.

Тип отделяется от имени двоеточием. Несколько параметров можно объединить в группу с общим описанием типа. В качестве разделителя в такой группе используется запятая. В заголовке подпрограммы нельзя вводить новый тип.

Каждый такой параметр является локальным по отношению к описываемой процедуре, к нему можно обращаться только в пределах данной процедуры.

Фактические параметры – это параметры, которые передаются при обращении к ней. Число и тип формальных и фактических параметров должны совпадать с точностью до их следования. Параметры подпрограмм могут быть параметрами-значениями и параметрами-переменными.

Параметры-значения – это когда копия фактического параметра становится значением соответствующего формального параметра. Внутри подпрограммы можно производить любые действия с данным формальным параметром (допустимым для его типа), но эти изменения никак не отражаются на значении фактического параметра, то есть каким он был до вызова процедуры, то таким же и останется после ее завершения.

В качестве фактического параметра на месте параметра-значения при вызове подпрограммы может выступать любое выражение совместимого для присваивания типа

Параметры-переменные – это передача параметров по ссылке. В списке перед такими формальными параметрами должен стоять идентификатор *var*.

Действие слова *var* распространяется до ближайшей точки с запятой, т.е. в пределах одной группы.

Передается адрес фактического параметра (обязательно переменной), после этого формальный параметр становится его синонимом. Любые операции с формальным параметром выполняются непосредственно над фактическим параметром.

Список параметров может быть сколь угодно длинным, хотя трансляторы обычно накладывают определенные ограничения в зависимости от размера стека.

Правила структурного программирования относительно использования подпрограмм:

- Каждая процедура должна иметь одну точку входа и одну точку выхода.
- Использование глобальных переменных в процедуре должно быть минимально.

Взаимодействие вызывающей логики с процедурой должно осуществляться только через параметры заголовка процедуры.

Обращение к подпрограмме или вызов является одной из наиболее используемых операций, поэтому он реализуется аппаратно. Подпрограмма обычно находится в отдельном месте памяти и для её вызова необходимо, во-первых, сообщить процессору о том, что следующей будет первая команда подпрограммы (передать процессору адрес первой команды подпрограммы), а во-вторых, запомнить, где находится команда, которая будет выполняться после окончания подпрограммы.

Возвращение из подпрограммы основано на механизме использования стека. Во время каждого вызова подпрограммы в стек заносится адрес возвращения. В стек также заносятся значения параметров для подпрограммы, а также резервируется место для всех внутренних (локальных) переменных подпрограммы. Это позволяет подпрограмме вызывать саму себя с новыми значениями параметров – выполнять рекурсивный вызов подпрограммы.

Для стека присущ принцип «первый пришел – последний ушел». За этим принципом данные, которые пополняют стек, прячут под собой то, что было положено раньше. Фактически операции со стеком осуществляются только через операции с верхним элементом, а именно *push* – забрать из стека и *pop* – положить в стек.

Процедура - это подпрограмма для выполнения какой-то законченной последовательности действий. Любая процедура начинается обязательно с заголовка. Он состоит из зарезервированного слова *procedure*, за которым следует идентификатор имени процедуры, а далее в круглых скобках список формальных параметров:

```
procedure <имя процедуры> (<список формальных параметров>);
```

За заголовком могут идти такие же разделы, что и в основной программе. В отличие от основной программы процедура завершается не точкой, а точкой с запятой.

Пример. Процедура ввода N-чисел.

Пусть в основной программе определен тип:

```
type tarr: array[1..100] of integer;
```

Процедура может иметь вид:

```
procedure inpint(var mas: tarr; n: integer);
```

```
var i: integer;  
begin  
write('Введи ', n, ' чисел');  
for i:=1 to n do  
read (ms[i])  
end;
```

Для вызова процедуры из основной программы или другой подпрограммы следует записать оператор, состоящий из имени процедуры и списка фактических параметров, которые должны совпадать по количеству и типам с формальными параметрами процедуры. Например:

```
inpint(m,k);
```

означает, что вызывается процедура *inpint* для ввода *k* целых чисел в массив *m*. Естественно, что в этом случае параметр *k* целого типа, а *m* – массив типа *tarr*.

При вызове процедуры линейный ход выполнения основной программы становится нелинейным – управление вычислительным процессом передается на участок программного кода, занимаемой процедурой. После выполнения процедуры осуществляется возврат на оператор основной программы, следующий за вызовом процедуры.

Функция – это подпрограмма, предназначенная для вычисления какого-либо параметра

Заголовок функции состоит из слова *function*, за которым следует имя функции, далее в круглых скобках – список формальных параметров, затем через двоеточие записывается тип функции – тип возвращаемого параметра. В теле функции хотя бы раз имени функции должно быть присвоено значение.

Пример. Функция вычисления факториала числа N.

```
function factorial(n:byte): longint;
```

```
var fact: longint;
```

```
i: byte;
```

```
begin  
fact:=n;  
for i:=n-1 downto 2 do  
fact:=fact*i;  
factorial:=tact
```

end;

Если имя функции внутри ее описания используется не в левой части оператора присваивания, то это означает, что функция вызывает себя рекурсивно.

Для вызова функции из основной программы или другой подпрограммы следует в выражении, где необходимо использовать значение функции, указать имя функции со списком фактических параметров, которые должны совпадать по количеству и типам с формальными параметрами.

Пример рекурсивной функции вычисления факториала числа N .

```
function factorial(n: integer): longint;
```

```
begin
```

```
if n=1 then factorial:=1
```

```
else factorial:=n*factorial(n-1)
```

```
end;
```

Для досрочного выхода из подпрограммы может быть использована функция *exit*.

8.2. Задание для контроля знаний

Составить программу на языке Паскаль для вычисления по заданным формулам с использованием пользовательской функции $f(a,b,i)$. Параметры a, b считать вещественными, i – целым.

Пример решения задания 8.2.

Исходные математические формулы

$$f(a,b,i) = \begin{cases} \frac{(3a-b)i^2 - b}{\sqrt{|3a-2bi| + i^2}} \text{ npu } a \geq \frac{2}{3}b \\ \frac{(2ab-3i)^2 + 5}{\sqrt{a^2i^2 - (3a-2b)i + b^2}} \text{ npu } a < \frac{2}{3}b \end{cases}$$

Программа на языке Паскаль

```
program prim;
```

```
var
```

```
a,b: real;
```

```
i: integer;
```

```
function f(a,b:real;i:integer):real;
```

```
begin
```

```
if a>=(2*b/3) then
```

```
f:=((3*a-b)*sqr(i)-b)/sqrt(abs(3*a-2*b*i)+sqr(i))
```

```
else
```

```
f:=(sqr(2*a*b-3*i)+5)/sqrt(sqr(a*i)-(3*a-2*b)*i+sqr(b))
```

```
end;
```

```
begin
```

```
a:=1.0;
```

```
b:=2.0;
```

```
i:=3;
```

```
writeln(f(a,b,i):12:5);
```

```
end.
```

Варианты задания 8.2.

- | | |
|--|---|
| <p>1) $\begin{cases} \frac{(i+1)(2a+bi)^{\frac{3}{2}}}{(b-a)^2 + ai + i^2} \text{ npu } a > b \\ \frac{(i-1)(2b+a)^{\frac{3}{2}}}{(b-a)^2 + ai + i^2} \text{ npu } a \leq b \end{cases}$</p> | <p>2) $\begin{cases} \frac{(a-bi)^4}{i(a^2 + 3bi + 4i^2)} \text{ npu } b > 1 \\ \frac{(a+bi)^4}{i(a^2 + 3(1-b)i + 4i^2)} \text{ npu } b \leq 1 \end{cases}$</p> |
| <p>3) $\begin{cases} \frac{(a+(-1)^i bi)^2}{i\sqrt{i+b^2}} \text{ npu } b > 0 \\ \frac{(a+bi)^2}{i\sqrt{i+b^2}} \text{ npu } b \leq 0 \end{cases}$</p> | <p>4) $\begin{cases} \frac{ia^2 + (-1)^{i+1}i^2(b-a)}{\sqrt{5a^2 + ai + b^2i^2}} \text{ npu } a > 0 \\ \frac{ia^2 + i^2(b-a)}{\sqrt{5a^2 - ai + b^2i^2}} \text{ npu } a \leq 0 \end{cases}$</p> |
| <p>5) $\begin{cases} \frac{2ai^2 - bi + 3ab}{\sqrt{5a^2 + 4b^2 + i^2}} \text{ npu } a < b \\ \frac{i^2 + 1}{a-b+i} \text{ npu } a \geq b \end{cases}$</p> | <p>6) $\begin{cases} ia^2 - \frac{i^2 + b}{i + a^2} \text{ npu } a > b \\ \left(ia^2 + \frac{b}{i} \right)^2 \text{ npu } a \leq b \end{cases}$</p> |

$$\begin{array}{ll}
7) \left\{ \begin{array}{l} \frac{2(ab)^i + i^2 - a}{2(a^2 + b^2) + i} \text{ при } |ab| \leq 1 \\ \frac{2ab + (-1)^i(a-i)}{2(a^2 + b^2)} \text{ при } |ab| > 1 \end{array} \right. & 8) \left\{ \begin{array}{l} \frac{a + b + i + (-1)^i(b-a)}{\sqrt{ai}} \text{ при } a > 0 \\ |a+i| + |b-i| \text{ при } a \leq 0 \end{array} \right. \\
9) \left\{ \begin{array}{l} \frac{(a-i)^2 + (b+i)^2}{\sqrt{a^2 + 2b^2 + i^2}} \text{ при } a \leq b \\ \frac{(a+i)^2 + (b-i)^2}{\sqrt{a^2 + 2b^2 + i^2}} \text{ при } a > b \end{array} \right. & 10) \left\{ \begin{array}{l} \frac{(ai + 2b)^2 + (-1)^i i}{\sqrt{i^2 + i}} \text{ при } a > 0 \\ \frac{(ai + 2b)^2 + i}{\sqrt{i^2 - i + 1}} \text{ при } a \leq 0 \end{array} \right. \\
11) \left\{ \begin{array}{l} \frac{(2a + 3i)^2 - (3a - 2i)^2}{\sqrt{a^2 + (-1)^i ai + b^2 + 1}} \text{ при } a < 2b \\ \frac{(3i - 2a)^2 + (2i - 3a)^2}{\sqrt{a^2 i^2 - 2abi + b^2 + 1}} \text{ при } a \geq 2b \end{array} \right. & 12) \left\{ \begin{array}{l} \frac{(a^2 - b)i^2 + 3ai + b}{\sqrt{(3ai - 2)^2 + 4b^2}} \text{ при } |ab| \geq 1 \\ \frac{(ai - b)^2 - (ab)^i}{2i + ab} \text{ при } |ab| < 1 \end{array} \right. \\
13) \left\{ \begin{array}{l} \frac{(a^2 - b)i^2 + 3ai + b}{\sqrt{(2ai + b)^2 - bi^2}} \text{ при } b \leq 0 \\ \frac{a + i + \sqrt{bi^2 + 1}}{a^2 + bi} \text{ при } b > 0 \end{array} \right. & 14) \left\{ \begin{array}{l} \frac{(3a - b)i^2 - b}{\sqrt{|3a - 2bi| + i^2}} \text{ при } a \geq \frac{2}{3}b \\ \frac{(2ab - 3i)^2 + 5}{\sqrt{a^2 i^2 - (3a - 2b)i + b^2}} \text{ при } a < \frac{2}{3}b \end{array} \right.
\end{array}$$

Если количество индексов, необходимых для доступа к элементам массива равно n , то такой массив называется n -мерным. Количество индексов еще называется размерностью массива.

В качестве имени массива может быть использован любой идентификатор. Обращение к элементам массива осуществляется с помощью указания имени массива и списка индексов в квадратных скобках. В качестве разделителя в списке индексов используется запятая.

Пример.

`M[1,10]`

`M[1][10]`

В общем случае каждый индекс компоненты массива может быть задан выражением соответствующего типа.

Объявление массива в программе

В программе массив может быть объявлен как

- Тип
- Константа
- Переменная

Чтобы задать тип-массив, используется зарезервированное слово `array`, после которого следует указать тип индексов компонент (в квадратных скобках) и далее после слова `of` – тип самих компонент. Индексы в массиве могут быть любого порядкового типа, кроме типа `LongInt`. Размерность массива может быть любой. Компоненты массива могут быть любого, в том числе и структурированного, типа.

Для двумерного массива считают, что первый индекс обозначает номер строки, а второй – номер столбца воображаемой таблицы, элементами которой являются соответствующие значения данного массива. Такая логическая структура массива еще называется прямоугольной матрицей. Каждый элемент матрицы однозначно определяется указанием номера строки и номера столбца.

Пусть константы $MaxN$ и $MaxM$ обозначают количество строк и столбцов двумерного массива, т.е. они описаны в программе следующим образом

`Const`

`MaxN = 10;`

`MaxM = 15;`

9. ЛЕКЦИЯ № 9

РЕКУРСИЯ И РЕКУРСИВНЫЕ АЛГОРИТМЫ РЕКУРСИИ. ПРИМЕРЫ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ РЕКУРСИИ. ПЕРЕБОР С ВОЗВРАТАМИ.

9.1. Двумерные массивы

Набор однотипных данных, доступ к элементам которого осуществляется с помощью двух индексов, называется двумерным массивом.

Массивы, положение элементов в которых описывается двумя индексами, называют двумерными.

Объявление массива как типа:

Type

OmyArray = *array[1..MaxM] of integer*; (Одномерный массив из целых чисел).

TmyArray = *array[1..MaxN] of OmyArray*; (Одномерный массив, элементами которого являются одномерные массивы из целых чисел).

или

TmyArray = *array[1..MaxN, 1..MaxM] of integer*; (Двумерный массив из целых чисел)

При задании значений константе массиву компоненты указываются в круглых скобках и разделяются запятыми, причем, если массив многомерный, внешние круглые скобки соответствуют левому индексу, вложенные в них круглые скобки – следующему индексу и т.д.

Пример

Type

Arr = *array[1..3, 1..2] of integer*;

Const

Mat: Matrix = ((1,2), (3,4), (5,6));

Последняя константа соответствует следующей структуре:

1	2
3	4
5	6

Объявление массива как структурированной переменной

Var

Mas: *array[1..3, 1..2] of real*;

Количество элементов двумерного массива Количество элементов в двумерном массиве определяется путем умножения числа строк на число столбцов этого массива.

Пример

Var

A: *array[-1..1, -5..5] of integer*;

В этом случае – количество строк – 3, столбцов 11. Следовательно массив А содержит 33 элемента.

Объем памяти, необходимый для хранения массива Количество необходимой памяти для хранения массива определяется путем умножения числа элементов массива на число байт, требуемое для хранения отдельной компоненты массива.

Пример

Var

A: *array[-1..1, -5..5] of integer*;

Для хранения этого массива необходимо

33 x 2 байта = 66 байт памяти

Как хранятся элементы двумерного массива в памяти компьютера В памяти компьютера массив хранится построчно.

Формула для определения адреса ячейки памяти, где хранится i,j элемент двумерного массива

$Addr(a[i,j]) = addr(a[1,1]) + v * (M * (i-1) + j - 1)$, где

M – количество столбцов

V – число байт, необходимых для хранения одного элемента массива.

Для организации доступа ко всем элементам двумерного массива необходимо использовать вложенные циклы.

Вид последовательности доступа к различным элементам двумерного массива зависит от организации этих вложенных циклов.

Одному массиву можно присвоить значение другого массива, но только идентичного типа.

Пример, если

Var a, b: *array [1..5] of integer*;

C: *array[1..5] of integer*;

то оператор *a:=b*; - допустим, а оператор *a:=c*; - нет.

Оператор *a:=b*; эквивалентен следующей цепочке операторов

For i:=1 to 5 do

For j:=1 to 5 do

a[i,j]:=b[i,j];

Однотипные массивы можно использовать в логических отношениях равенства (=) и неравенства (<>). Другие операции отношения применяются только к отдельным элементам массива.

В Турбо-Паскале есть ограничения по памяти, отводимой под любую переменную. Это ограничение составляет 64 Кбайта.

9.2. Задание для контроля знаний

Для заданных границ интегрирования a и b вычислите значение определенного интеграла (n, m - целые числа, p - вещественное).

Пример решения задания 9.2.

Исходные математические формулы

$$\int x^m \cos px dx = \begin{cases} \frac{x^m}{p} \sin px + \frac{m}{p} \int x^{m-1} \sin px dx, & m \geq 1 \\ \frac{\sin px}{p}, & m = 0 \end{cases}$$

$$\int x^m \sin px dx = \begin{cases} -\frac{x^m}{p} \cos px + \frac{m}{p} \int x^{m-1} \cos px dx, & m \geq 1 \\ -\frac{\cos px}{p}, & m = 0 \end{cases}$$

Программа на языке Паскаль

<pre> program prim; var p,a,b: real; k:integer; function integrs(m:integer):real; forward; function integrc(m:integer):real; var am,bm:real; i: integer; begin if m>0 then begin am:=1.0; for i:=1 to m do am:=am*a; bm:=1.0; for i:=1 to m do bm:=bm*b; integrc:=(integrs(m-1)*m/p + (bm*sin(p*b)-am*sin(p*a))/p; </pre>	<pre> function integrs(m:integer):real; var am,bm:real; i: integer; begin if m>0 then begin am:=1.0; for i:=1 to m do am:=am*a; bm:=1.0; for i:=1 to m do bm:=bm*b; integrs:=(integrs(m-1)*m/p + (am*cos(p*a)-bm*cos(p*b))/p; end else integrs:=(cos(p*a)-cos(p*b))/p; end; begin a:=0.0; </pre>
--	---

<pre> end else integrc:=(sin(p*b)-sin(p*a))/p; end; </pre>	<pre> b:=pi/2; p:=2.0; k:=3; writeln(integrc(k):12:5); end. </pre>
--	--

Варианты задания 9.2.

- 1)
$$\int \sin^n x dx = \begin{cases} -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n} \int \sin^{n-2} x dx, & n > 2 \\ \frac{x}{2} - \frac{1}{4} \sin 2x, & n = 2 \\ -\cos x, & n = 1 \end{cases}$$
- 2)
$$\int \cos^n x dx = \begin{cases} \frac{\cos^{n-1} x \sin x}{n} + \frac{n-1}{n} \int \cos^{n-2} x dx, & n > 2 \\ \frac{x}{2} + \frac{1}{4} \sin 2x, & n = 2 \\ \sin x, & n = 1 \end{cases}$$
- 3)
$$\int \frac{dx}{\sin^n x} = \begin{cases} -\frac{1}{n-1} \frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\sin^{n-2} x}, & n \geq 2 \\ \ln \operatorname{tg} \frac{x}{2}, & n = 1 \\ x, & n = 0 \end{cases}$$
- 4)
$$\int \frac{dx}{\cos^n x} = \begin{cases} \frac{1}{n-1} \frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\cos^{n-2} x}, & n \geq 2 \\ \ln \operatorname{tg} \left(\frac{\pi}{4} + \frac{x}{2} \right), & n = 1 \\ x, & n = 0 \end{cases}$$

$$5) \int x^n e^{px} dx = \begin{cases} \frac{x^n e^{px}}{p} - \frac{n}{p} \int x^{n-1} e^{px} dx, n > 1 \\ \frac{e^{px}}{p^2} (px - 1), n = 1 \end{cases}$$

$$6) \int x^n p^{mx} dx = \begin{cases} \frac{x^n p^{mx}}{n \ln p} - \frac{n}{m \ln p} \int x^{n-1} p^{mx} dx, n > 1 \\ \frac{x p^{mx}}{m \ln p} - \frac{p^{mx}}{m(\ln p)^2}, n = 1 \end{cases}$$

$$7) \int e^{px} \cos^n x dx = \begin{cases} \frac{e^{px} \cos^{n-1} x (p \cos x + n \sin x)}{p^2 + n^2} + \\ + \frac{n(n-1)}{p^2 + n^2} \int e^{px} \cos^{n-2} x dx, n \geq 2 \\ - \frac{e^{px} (\sin x + p \cos x)}{p^2 + 1}, n = 1 \\ \frac{1}{p}, n = 0 \end{cases}$$

$$8) \int e^{px} \sin^n qx dx = \begin{cases} \frac{e^{px} \sin^{n-1} qx (p \sin x - nq \sin qx)}{p^2 + n^2 q^2} + \\ + \frac{n(n-1)q^2}{p^2 + n^2 q^2} \int e^{px} \sin^{n-2} qx dx, n \geq 2 \\ - \frac{e^{px} (p \sin qx - q \cos qx)}{p^2 + q^2}, n = 1 \\ \frac{1}{p}, n = 0 \end{cases}$$

$$9) \int \ln^n x dx = \begin{cases} x \ln^n x - n \int \ln^{n-1} x dx, n > 1 \\ x \ln x - x, n = 1 \end{cases}$$

$$10) \int x^m \ln^n x dx = \begin{cases} \frac{x^{m+1}}{m+1} \ln^n x - \frac{n}{m+1} \int x^m \ln^{n-1} x dx, n > 1 \\ x^{m+1} \left[\frac{\ln x}{m+1} - \frac{1}{(m+1)^2} \right], n = 1 \end{cases}$$

$$11) \int \frac{dx}{(p^2 + x^2)^n} = \begin{cases} \frac{1}{2(n-1)p^2} \left[\frac{x}{(p^2 + x^2)^{n-1}} + (2n-3) \int \frac{dx}{(p^2 + x^2)^{n-1}} \right], n > 1 \\ \frac{1}{p} \operatorname{arctg} \frac{x}{p}, n = 1 \end{cases}$$

$$12) \int \operatorname{tg}^n x dx = \begin{cases} \frac{\operatorname{tg}^{n-1} x}{n-1} - \int \operatorname{tg}^{n-2} x dx, n \geq 2 \\ -\ln|\cos x|, n = 1 \\ x, n = 0 \end{cases}$$

$$13) \int \operatorname{ctg}^n x dx = \begin{cases} -\frac{\operatorname{ctg}^{n-1} x}{n-1} - \int \operatorname{ctg}^{n-2} x dx, n \geq 2 \\ -\ln|\sin x|, n = 1 \\ x, n = 0 \end{cases}$$

$$14) \int x^m \sin px dx = \begin{cases} -\frac{x^m}{p} \cos px + \frac{m}{p} \int x^{m-1} \cos px dx, m \geq 1 \\ -\frac{\cos px}{p}, m = 0 \end{cases}$$

10. ЛЕКЦИЯ № 10

ПОИСК И СОРТИРОВКА ПОСЛЕДОВАТЕЛЬНЫЙ ПОИСК. ДВОИЧНЫЙ ПОИСК. СОРТИРОВКА ОБМЕНОМ. СОСОРТИРОВКА ВЫБОРОМ. МЕТОД ПРОСТЫХ ВСТАВОК.

10.1. Множества

Напомним два отличительных свойства множества в математике: - каждый элемент множества включается в него лишь один раз; - элементы в

множестве не упорядочены. Фундаментальный тип данных, который называется множественным типом данных (или просто типом множество) позволяет описать множество всех подмножеств множества некоторого заданного типа, называемого базовым.

Описание типа множество имеет следующий синтаксис:

set of <базовый тип> ,

где <базовый тип> - это любой порядковый тип, количество значений которого не более 256.

Например, тип Integer не может быть базовым типом, однако его поддиапазон 1..255 - может.

Для инициализации переменной типа множество часто используется оператор присваивания, в правой части которого находится конструктор множества - список его элементов, заключённый в квадратные скобки.

Каждый элемент списка представляет собой выражение со значением базового типа или диапазон значений базового типа.

Операции над переменными типа множество При работе с множествами разрешается использование следующих: (1) бинарных операций: объединение множеств (+), пересечение множеств (*), разность множеств (-); (2) предикатов: равенство множеств (=), неравенство множеств (<>), включение множеств (>=), обратное включение множеств (<=), принадлежность элемента множеству (*in*).

Предикат *in* имеет следующий синтаксис:

<элемент> *in* <множество>

Семантика предиката *in* - "обычная", т.е. его значением является *true*, если значение аргумента <элемент> содержится в значении параметра <множество>, и *false* - в противном случае.

Перечислим операции над множествами в порядке убывания приоритета:

Приоритет	1	2	2	3	3	3	3	4
Операции над множествами	*	+	-	=	<>	>=	<=	<i>in</i>

Пример. Построение по заданным множествам символьного типа Y1 и Y2 нового множества (Y1UY2) и вывод на экран его элементов.

program prim;

```

type sett=set of char;
var y1,y2: sett; { заданные множества }
x : sett; { результат }
procedure new_set (y1,y2: sett; var x: sett);
{построение множества x из множеств y1 и y2}
begin
x:=(y1*y2)+(y1-y2)
end;
procedure output_set (x: sett);
{Процедура вывода на экран элементов множества x}
var c: char;
begin
for c:='a' to 'r' do if c in x then write(c,' ')
end;
begin
y1:=['a','b','d','r','m'];
y2:=['r','a','h','d','b','m','l'];
new_set(y1,y2,x);
write('Множество x состоит из следующих элементов: ');
output_set(x);
writeln;
writeln('Проверим включение y1 в y2: ',y1<=y2);
end.

```

10.2. Задание для контроля знаний

Варианты задания 10.2.

Вычислите значение

- | | |
|--------------------------------------|------------------------------------|
| 1) отношения [2]=[2,2,2] | 2) отношения ['c','b']=['c'..'b'] |
| 3) отношения ['a','b']=['b','a'] | 4) отношения [2,3,5,7]<=[1..9] |
| 5) отношения [4,5,6]=[4..6] | 6) отношения [3,6..8]<=[2..7,9] |
| 7) отношения []<=['0'..'9'] | 8) отношения 'q' in ['a'..'z'] |
| 9) отношения trunc(3.9) in [1,3,5,7] | 10) отношения odd(4) in ['a'..'z'] |
| 11) отношения [2]<[1..3] | 12) отношения 66=[66] |

- | | |
|---|--|
| 13) выражения [1,3,5]+[2,4] | 14) выражения [1,3,5]*[2,4] |
| 15) выражения [1,3,5]-[2,4] | 16) выражения [1..6]+[3..8] |
| 17) выражения [1..6]*[3..8] | 18) выражения [2,4]-[1..5] |
| 19) выражения []+[4,5,6] | 20) выражения []*[4] |
| 21) выражения []-[4] | 22) выражения [2..13]*[3,13..60]+[4..10]-[5..15]*[6] |
| 22) выражения [2..10]-[4,6]-[2..12]*[8..15] | 24) выражения ('0'..'7')+['2'..'9']*(['a']+['z']) |

11. ЛЕКЦИЯ № 11

СРЕДСТВА ГРАФИКИ ТЕКСТОВЫЙ И ГРАФИЧЕСКИЙ РЕЖИМЫ. ГРАФИЧЕСКИЕ КООРДИНАТЫ. КОМАНДЫ ГРАФИЧЕСКОГО РЕЖИМА. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ В ДЕКАРТОВЫХ КООРДИНАТАХ. ПРИМЕНЕНИЕ ПОЛЯРНЫХ КООРДИНАТ. ПОСТРОЕНИЕ ДВИЖУЩИХСЯ ИЗОБРАЖЕНИЙ.

11.1. Работа с символьными строками

Для работы с символьной информацией принято использовать структуру данных - строка. В языке Паскаль её называют стрингом (от англ. string - строка).

Определение переменных типа *String* содержит служебное слово *String*, за которым идет максимальная длина строки в квадратных скобках (целочисленная константа в диапазоне от 0 до 255).

Например: `var a : String [14]; line: String [80].`

Доступ к элементам строки производится с помощью индексации. В результате получается величина типа *Char*. Например: `A[6]`. Тип *String* и стандартный тип *Char* совместимы, т.е. строки и символы могут употребляться в одних и тех же выражениях. При сравнении двух строк в результате возвращается *true* только в том случае, когда сравниваемые строки совпадают посимвольно и имеют одинаковую длину (принадлежат одному и тому же типу).

Синтаксис и семантика функций для работы со строками

Функция	Назначение	Тип аргумента	Тип результата
---------	------------	---------------	----------------

<i>length(s)</i>	Возвращает длину строки <i>s</i>	<i>string</i>	<i>integer</i>
<i>concat(s1,...sn)</i>	Возвращает строку, полученную конкатенацией строк <i>s1,s2,...sn</i> . Длина результата не должна превосходить 255	<i>string</i>	<i>string</i>
<i>copy (s,pos,len)</i>	Возвращает строку, полученную из <i>len</i> символов строки <i>s</i> , начиная с позиции <i>pos</i> . Если <i>pos</i> больше, чем длина строки <i>s</i> , то возвращается пустая строка. Если <i>pos+len</i> больше длины строки <i>s</i> , то возвращаются только символы, принадлежащие строке <i>s</i>	<i>s: string</i> <i>pos:integer</i> <i>len: integer</i>	<i>string</i>
<i>pos(pattern, source)</i>	Возвращает номер символа, начиная с которого <i>pattern</i> входит в <i>source</i>	<i>string</i>	<i>integer</i>

В языке Паскаль для работы с символьной информацией существуют стандартные процедуры.

Процедура *str*.

Синтаксис процедуры:

`str(value,st)`, где: *value* - параметр, имеющий тип *Integer* или *real*, *st* - переменная типа *string*.

Процедура преобразует значение *value* в строковое и помещает результат в переменную *st*. Например, если *i=1234*, то в результате выполнения процедуры `str(i,st)` получим, что значением переменной *st*

является '1234'; если же $x=2.5E4$, то в результате выполнения процедуры $str(x,st)$ получим, что значением переменной st является '25000'.

Процедура val.

Синтаксис процедуры:

$val(st,var,code)$, где: var - параметр типа *integer* или *real*, $code$ - параметр типа *integer*, а st - переменная типа *string*.

Процедура преобразует строковое выражение st в величину типа *integer* или *real* (соответственно типу var) и помещает значение в var . Выражение не должно содержать символов "пробел". Переменная $code$ должна быть типа *integer*. Если при преобразовании не обнаружено ошибок, то значение $code$ равно 0, в противном случае значение $code$ равно номеру позиции, в которой обнаружена первая ошибка. Например:

(1) если $st='234'$, то в результате обращения к процедуре $val(st,i,res)$ получим, что значением переменной i является 234, а $res=0$;

(2) если $st='12x'$, то в результате обращения к процедуре $val(st,i,res)$ получим, что значение переменной i не определено, а $res=3$;

(3) если же $st='2.5e4'$, то в результате обращения к процедуре $val(st,i,res)$ получим, что значением переменной i является 25000, а $res=0$.

11.2. Задание для контроля знаний

Пример решения задания 11.2.

Написать программу, учитывающую число вхождений подслова "mu" в заданное слово.

```
program prim;
var x: string[60]; { исходное слово }
    i, { параметр цикла }
    s: byte; { счетчик }
begin
s:=0;
write('Введите исходное слово: ');
readln(x);
for i:=1 to length(x)-1 do if copy(x,i,2)='mu' then s:=s+1;
writeln('Число вхождений подслова "mu": ',s)
end.
```

Варианты задания 11.2.

Напишите программу

- 1) замены в слове X всех букв "a" на сочетание "ку";
- 2) записывающую слово X в обратном порядке;
- 3) замены всех сочетаний "ку" в слове X на букву "a";
- 4) выделяющую в слове X все буквы "о" символами "пробел". Например, слово "сосна" должно быть преобразовано в "с о сна";
- 5) удваивающую каждую букву слова X;
- 6) выделяющую в слове X каждую букву "о" с помощью буквы "-" слева и справа;
- 7) для вычеркивания всех букв "о", стоящих в слове X на чётных местах;
- 8) для вычеркивания из слова X всех букв "К" и "G";
- 9) для вычеркивания в слове X всех букв, стоящих на нечётных местах после буквы "a";
- 10) для вычеркивания из слова X всех букв "р", перед которыми стоит буква "a";
- 11) для вычеркивания из слова X каждой третьей буквы;
- 12) проверяющую, есть ли в слове X две одинаковые буквы;
- 13) выясняющую, есть ли в слове X буква "a", стоящая на нечётном месте после буквы "к";
- 14) проверяющую, есть ли в слове X буква "к", стоящая на чётных местах перед буквой "и";
- 15) проверяющую, все ли буквы слова X одинаковы;
- 16) выясняющую, можно ли из букв слова X составить слово Y;
- 17) для проверки, есть ли в слове X буквы "в". Если есть, то найдите номер первой из них;
- 18) выясняющую, есть ли в слове X буква "к", и, если есть, то замените все буквы "a" в этом слове на "с";
- 19) для подсчёта числа букв "о", стоящих в слове X на чётных местах;
- 20) для подсчёта числа сочетаний "ку" в слове X;
- 21) для подсчёта суммы мест, на которых в слове X стоит буква "б";
- 22) для подсчёта в слове X всех сочетаний "нн". Считайте, что в слове "ннн" подслово "нн" встречается один раз;

- 23) выясняющую, сколько раз в слове X встречается сочетание из первых двух букв слова Y;
- 24) выясняющую, какая из букв (первая или последняя) встречается в слове X чаще;
- 25) для подсчёта числа букв "a" в слове X, стоящих на местах, номер которых кратен трем;

12. ЛЕКЦИЯ № 12

ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ ОБЪЕКТЫ. ИНКАПСУЛЯЦИЯ. НАСЛЕДОВАНИЕ. ВИРТУАЛЬНЫЕ МЕТОДЫ. ДИНАМИЧЕСКОЕ СОЗДАНИЕ ОБЪЕКТОВ. ПОЛИМОРФИЗМ.

12.1. Файлы

Файл - это последовательность записей, размещаемая на внешних запоминающих устройствах (внешней памяти) и рассматриваемая в процессе пересылки и обработки как единое целое. Таким образом, под файлом понимается "физическое" место на диске, имеющее имя и размер.

Файл - это совокупность упорядоченных и взаимосвязанных записей, имеющая описание для идентификации отдельных записей, таким образом, файл - это структура данных.

Структура данных файл в зависимости от типа элементов делится на типизированные, текстовые и нетипизированные файлы.

Типизированный файл - это файл, состоящий из последовательности элементов (называемых компонентами) одного типа. Количество компонентов файла при задании файла не ограничивается.

Так как все компоненты файла имеют одинаковую "длину", то позиция каждого компонента в файле может быть вычислена.

Длиной файла называется число находящихся в нём компонентов. Пустым файлом называется файл, не содержащий компонентов. Длина пустого файла равна нулю. Общий вид описания файлового типа:

$type\ r=file\ of\ tc,$

где: *file* (от англ. файл), *of* (от англ. из) - служебные слова;

r - идентификатор типа;

tc - тип компонентов (не может быть типом *file*).

Отметим, что не допускается использование файловых переменных в операторах присваивания и в выражениях.

В каждый момент времени доступен только один компонент файла (при этом говорят, что на этот компонент установлен указатель файла). Следовательно, читать файл можно только последовательно по одному компоненту. Поэтому рассматриваемые файлы называются файлами последовательного доступа (или последовательными файлами).

Над файлами можно выполнять два явных вида действий:

Создание файла. Оно выполняется в результате добавления нового компонента в конец первоначально пустого файла. В процессе создания новые значения разрешается записывать только в конец файла;

Просмотр файла. Он выполняется в результате последовательного продвижения по файлу, начиная с его начала. При этом в каждый момент времени доступен лишь один компонент файла. В процессе просмотра файла изменять значения компонентов на новые запрещается.

Начать писать в файл можно только с самого его начала, дописывая новые компоненты последовательно одну за другой; для чтения также надо начинать просмотр файла с самого начала. Вследствие такой организации на одном просмотре файла нельзя совмещать и чтение, и запись информации: можно либо только читать из файла, либо только писать в файл. Однако, работая с различными файлами A и B, можно одновременно читать файл A и писать в файл B (и наоборот).

Все остальные действия над файлами последовательного доступа являются композицией его просмотра и создания.

Некоторые стандартные процедуры для последовательной работы с файлами

1. $assign\ (var\ f:\ file;\ name:\ string);\ assign\ (var\ f:\ file\ of\ type;\ name:\ string),$

где: *assign* (от англ. назначить) - имя процедуры;

f - имя файловой переменной (логическое имя файла);

name - имя файла на диске (физическое имя файла). Это имя связывается с файловой переменной *f*, и в дальнейшем все операции с *f* будут производиться над дисковым файлом с именем *name*. После выполнения этой процедуры *name* и *f* отождествляются, т.е. отождествляются логическое и физическое имена файла. Данная процедура должна выполняться перед использованием файла. Перед выполнением операций

ввода-вывода файл должен быть открыт с помощью одной из следующих процедур:

2. *reset (var f: file),*

где: *reset* (от англ. переустановить) - имя процедуры;

f - имя переменной файлового типа. Процедура открывает файл *f* для чтения и ставит указатель на начало первого компонента файла (на компонент с номером 0). Файловая переменная *f* должна соответствовать уже существующему на диске файлу, иначе возникает ошибка "I/O error" (от англ. *input/output error* - ошибка ввода-вывода).

3. *rewrite (var f: file),*

где: *rewrite* (от англ. установить на запись) - служебное слово;

f - имя переменной файлового типа.

Процедура открывает файл *f* для записи, «очищает» его и помещает указатель на начало первого компонента файла. Таким образом, существующий файл с данным именем будет уничтожен. Дискový файл, организуемый с помощью процедуры *rewrite*, создается пустым (не содержащим компонентов).

4. *read (var f: file of type; var v: type),*

где: *read* (от англ. Читать) – имя процедуры;

f – имя переменной файлового типа;

v – имя переменной (или список из нескольких переменных, разделенных запятыми), совпадающей по типу с типом *type* элементов файла.

Каждая переменная считывается с дискового файла, и после окончания чтения файловый указатель перемещается на следующий компонент файла.

5. *write (var f: file of type; var v: type),*

где: *write* (от англ. Писать) – имя процедуры;

f – имя переменной файлового типа;

v – имя переменной (или список нескольких переменных, разделенных запятыми), совпадающей по типу с типом элементов файла *type*.

Каждая переменная записывается в дискový файл, и после окончания операции записи файловый указатель перемещается на следующий компонент файла. После окончания работы с файлом, он должен быть закрыт, иначе результат будет потерян. «Закрывает» файл процедура *close*.

6. *close (var f: file),*

где: *close* (от англ. Закрыть) – служебное слово;

f – имя переменной файлового типа.

Дискový файл, отождествленный с переменной *F*, закрывается и в каталог диска вносятся необходимые изменения.

Процедуры для организации прямого доступа к компонентам файла последовательного доступа.

1. *seek (f, n),*

где: *seek* (от англ. Искать) – имя процедуры;

f – имя переменной файлового типа;

n – целочисленное выражение.

Процедура перемещает указатель файла на компоненту файла с номером *n* (номер первой компоненты файла равен 0).

2. *seek (f, filesize(f)),*

где: *seek* – имя процедуры;

f – имя переменной файлового типа;

filesize() – стандартная функция возвращает количество компонентов файла, а т.к. компоненты нумеруются, начиная с 0, то результат получается на единицу большим номера последнего компонента файла.

Процедура позволяет помещать указатель файла за конец файла.

Стандартные функции для работы с файлами.

1. *eof (var f: file): boolean,*

где: *eof* (от англ. *end of file* - конец файла) - имя функции;

F - имя переменной файлового типа.

Функция позволяет определить готовность файла к чтению, либо к записи информации. Если указатель файла продвинулся за конец файла (готовность к записи), то эта функция возвращает значение TRUE, в остальных случаях функция возвращает значение FALSE.

2. *filepos (var f: file of type): integer; filepos (var f: file): integer,*

где: *filepos* (от англ. *file position* - позиция файла) - имя функции;

F - имя переменной файлового типа.

Функция возвращает значение текущей позиции указателя файла. Номер первой компоненты файла равен 0.

3. *filesize (var f: file of type): integer; filesize (var f: file): integer,*

где: *filesize* (от англ. *file size* - размер файла) - имя функции;

F - имя переменной файлового типа.

Функция возвращает размер файла, т.е. число компонентов файла. Если $filesize(f)=0$, то файл является пустым.

Замечание. Функции $filepos()$ и $filesize()$ не применимы к текстовым файлам.

Текстовыми файлами (типа $text$) называются «почти» типизированные файлы, имеющими компоненты, которые могут иметь либо типа $char$, либо являться символом «Конец строки». Символ «Конец строки» – это совокупность символов «Перевод строки» с шестнадцатеричным кодом $0D$ и «Возврат каретки» с шестнадцатеричным кодом $0A$.

Символы в текстовом файле сформированы в последовательности символов типа $char$, которые называются строками; каждая строка оканчивается специальным составным символом, который называется «Конец строки».

Стандартные процедуры и функции для работы с текстовыми файлами.

1. $readln$ ($var f: text$)

Процедура осуществляет переход к началу следующей строки, т.е. пропускает все символы текущей строки.

2. $writeln$ ($var f: text$)

Процедура записывает символ "Конец строки" в текстовый файл.

3. $eoln$ ($var f: text$): $boolean$

Функция возвращает $true$, если в файле f найден символ "Конец строки" или символ "Конец файла" и $false$ - в противном случае.

Замечание. С символом $eoln$ оперируют следующие процедуры:

процедура $writeln(f)$ записывает символ "Конец строки" ($eoln$) в компонент файла, на который установлен указатель файла;

процедура $readln(f)$ пропускает оставшуюся часть текущей строки и устанавливает указатель файла на первый символ новой строки.

4. eof ($var f: text$): $boolean$

При работе с текстовым файлом функция возвращает результат $true$, если указатель файла расположен в позиции символа "Конец файла" (" $ctrl$ " + " z ") и $false$ - в противном случае.

5. $seekeoln$ ($var f: text$): $boolean$

Аналогична функции $eoln$, но пропускает символы "Пробел" перед проверкой на достижение символа "Конец строки".

6. $seekeof$ ($var f: file$): $boolean$

Аналогична функции EOF, но пропускает символы "Пробел" и символы "Конец строки" перед проверкой на достижение символа "Конец файла".

Нетипизированные файлы состоят из компонентов одинакового размера, структура которых неизвестна или не имеет значения. Такие файлы применяются, в частности, или в процедурах копирования, или при обработке файлов базы данных.

Описание нетипизированного файла: $var f: file$;

Для обработки файлов без типа применяют те же функции, что и для файлов с типом (т.е. функции eof , $filesize$, $filepos$).

Процедуры для обработки нетипизированных файлов.

1. $rewrite$ ($f: file$; $size: word$)

создает новый файл; если файл с таким именем уже существует на диске, то он удаляется и создается новый. Вторым параметром определяет размер записи в файле; если он отсутствует, то предполагается, что он равен 128 байтам.

2. $reset$ ($f: file$; $size: word$)

открывает существующий файл. В файл можно записывать информацию и читать из него. Вторым параметром определяет размер записи в файле; если он отсутствует, то предполагается, что он равен 128 байтам.

3. $blockread$ ($var f: file$; $var buf: ?$; $count: word$ [$var result: word$])

читает из файла F $Count$ -записей в буфер ввода buf ; если указан четвертый параметр, то он получает значение реального числа прочитанных записей. Количество прочитанных записей может не совпадать с $count$ при чтении последней, а потому, возможно, неполной, порции данных из файла. Если четвертый параметр не указан и количество затребованных и реально прочитанных записей не совпадает, то возникает ошибка ввода-вывода.

4. $blockwrite$ ($var f: file$; $var buf: ?$; $count: word$ [$var result: word$])

записывает в файл f $Count$ записей из буфера ввода buf ; если указан четвертый параметр, то он получает значение реального числа записанных компонент. Количество записанных записей может не совпадать с $count$ при нехватке места на диске. Если четвертый параметр не указан и количество требуемых и реально выведенных записей не совпадает, то возникает ошибка ввода-вывода.

5. $seek$ ($f: file$; $n: longint$)

передвигает указатель файла на n -ю запись (n - целочисленное выражение). Первый компонент имеет номер 0.

12.2. Задание для контроля знаний

Дан текстовый файл f. Получите все его строки, содержащие более 30 символов.

<pre> program prim; var f: text; s: string; begin assign(f, 'f.txt'); reset(f); while not eof(f) do </pre>	<pre> begin readln(f,s); if length(s)>30 then writeln(s); end; close(f); end. </pre>
--	---

Варианты задания 12.2.

- 1) Дан текстовый файл f. Запишите в "перевернутом" виде строки файла f в файл g. Порядок строк в файле g должен быть обратным по отношению к порядку строк исходного файла.
- 2) Дан текстовый файл. Перепишите в файл g все компоненты файла f с заменой в них символа "0" на символ "1" и наоборот.
- 3) Дан текстовый файл f. Получите самую длинную строку файла. Если в файле имеется несколько строк с наибольшей длиной, то получите одну из них.
- 4) Дан текстовый файл f. Запишите в "перевернутом" виде строки файла f в файл g. Порядок строк в файле g должен совпадать с порядком исходных строк в файле f.
- 5) Дан текстовый файл. Перепишите компоненты файла f в файл g, вставляя в начало каждой строки по одному пробелу. Порядок компонент должен быть сохранен.
- 6) Даны текстовый файл и строка s. Получите все строки файла f, содержащие в качестве фрагмента заданную строку s.
- 7) Дан текстовый файл f. Исключите пробелы, стоящие в концах его строк. Результат поместите в файл f1.
- 8) Дан файл f, компоненты которого являются действительными числами. Найдите: сумму компонент файла f.
- 9) Дан файл f, компоненты которого являются действительными числами. Найдите: произведение компонент файла f.

- 10) Дан файл f, компоненты которого являются действительными числами. Найдите: сумму квадратов компонент файла f.
- 11) Дан файл f, компоненты которого являются действительными числами. Найдите: модуль суммы и квадрат произведения компонент файла f.
- 12) Дан файл f, компоненты которого являются действительными числами. Найдите: последний компонент файла.
- 13) Дан файл f, компоненты которого являются действительными числами. Найдите: наибольший компонент.
- 14) Дан файл f, компоненты которого являются действительными числами. Найдите: наименьший компонент с четным номером.
- 15) Дан файл f, компоненты которого являются действительными числами. Найдите: наибольший модуль компонента с нечетным номером.
- 16) Дан файл f, компоненты которого являются действительными числами. Найдите: разность первого и последнего компонента файла.
- 17) Дан файл f, компоненты которого являются целыми числами. Найдите среди компонентов количество квадратов простых чисел.
- 18) Дано натуральное n. Запишите в файл g целые числа b_1, \dots, b_n , для которых $b_i = 2i + 3i + 1$, $i = 1, 2, \dots, n$.
- 19) Последовательность x_1, x_2, \dots образована по закону $x_i = (i - 0.1) / (i^3 + 2i)$, $i = 1, 2, \dots$. Дано действительное число $S > 0$. Запишите в файл N члены последовательности x_1, x_2, \dots , остановившись после первого члена последовательности, для которого выполнено условие $|x_i| < S$.
- 20) Дан символьный файл f. Получите копию файла в файле g.

13. ЛЕКЦИЯ № 13

ПРИКЛАДНЫЕ ВЫЧИСЛЕНИЯ РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ. ЗАДАЧИ ЛИНЕЙНОЙ АЛГЕБРЫ.

13.1. Записи

Комбинированный тип данных (записи)

Запись - тип данных, состоящий из фиксированного числа компонентов (называемых полями) одного или нескольких типов.

Приведём примеры описания типа запись:

```
type point=record
    x,y: real
end;
dates=record
    day : 1..31;
    mon : string[3];
    year: 1..3000
```

```
end;
var p,r: point;
dt: dates;
```

Можно определить массив записей, поля которых также являются массивами:

```
type student=array [1..n] of record
    fam : string[15];
    birth: dates;
    man : boolean;
    marks: array[1..10] of 0..5
```

```
end;
var group: student;
```

Идентификатор *group* можно использовать для хранения информации о группе студентов (фамилия, дата рождения, пол и оценки по 10 предметам).

Обращение к значению поля записи происходит при помощи составного имени, содержащего идентификатор переменной и имя поля, разделённые точкой. Например, *p.x*, *dt.mon*, *group[1].man*, *group[2].marks[1]*.

Составное имя может использоваться везде, где допустимо применение идентификатора типа поля: в выражениях, операторах ввода-вывода, присваивания, в качестве фактических параметров.

Обращение к полю записи с помощью составного имени может иметь громоздкий вид. Оператор *with*, решающий эту проблему, имеет следующий вид:

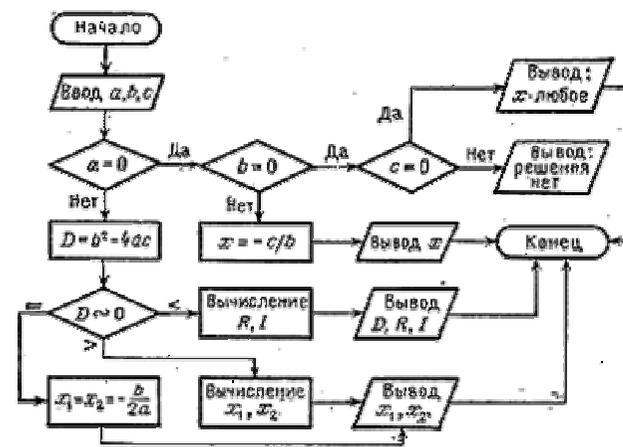
with <переменная типа запись> *do* <оператор>

Если после слова *with* задать имя записи, то в операторе, следующим за *do*, для доступа к полю можно указывать только имя поля без имени переменной.

13.2. Задание для контроля знаний

Пример решения задания 13.2.

По блок-схеме решения квадратного уравнения составить программу на языке Паскаль.



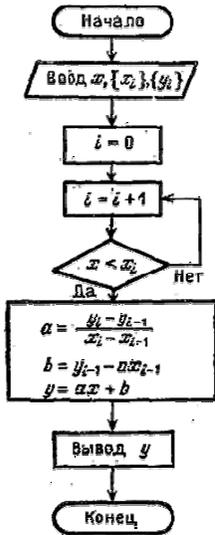
```
program root;
uses crt;
var
a,b,c,d,x,r,i,x1,x2: double;
begin
clrscr;
writeln('Решение квадратного
уравнения ax^2+bx+c=0');
write('Введите a = ');
read(a);
write('Введите b = ');
read(b);
write('Введите c = ');
read(c);
```

```
else begin
x:=-c/b;
writeln('x = ',x:8:3)
end
else begin
d:=b*b-4*a*c;
if d<0 then begin
r:=-b/2/a;
i:=sqrt(-d)/2/a;
writeln('x1 = ', r:8:3, ' + ', i:8:3,'i');
writeln('x2 = ', r:8:3, ' - ', i:8:3,'i');
end
else begin
x1:=(-b+sqrt(d))/2/a;
```

if $a=0$ then
 if $b=0$ then
 if $c=0$ then
 writeln('Корень может быть любым.')
 else
 writeln('Решений нет.')

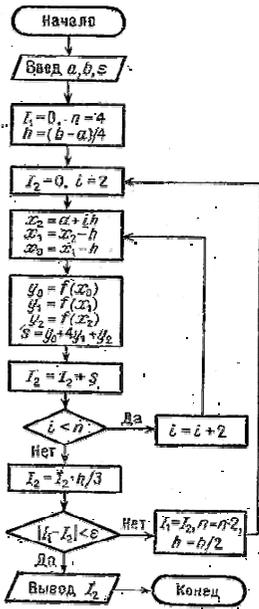
$x1 := -2 * c / (b + \text{sqrt}(d));$
 $x2 := c / (a * x1);$
 writeln('x1 = ', x1:8:3);
 writeln('x2 = ', x2:8:3);
 end;
 end;
 end.

Варианты задания 13.2.



1. Блок-схема линейной интерполяции.

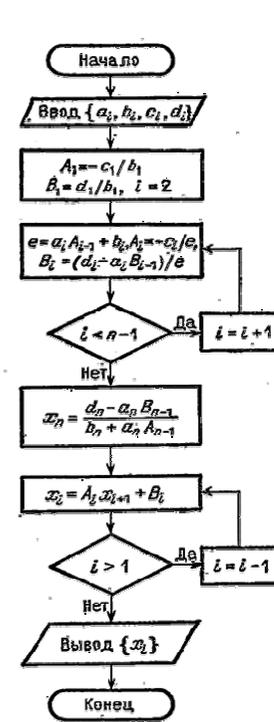
Линейная интерполяция состоит в том, что заданные точки (x_i, y_i) ($i = 0, \dots, n$) соединяются прямолинейными отрезками, и функция $f(x)$ приближенно заменяется ломаной с вершинами в данных точках.



2. Блок-схема метода Симпсона.

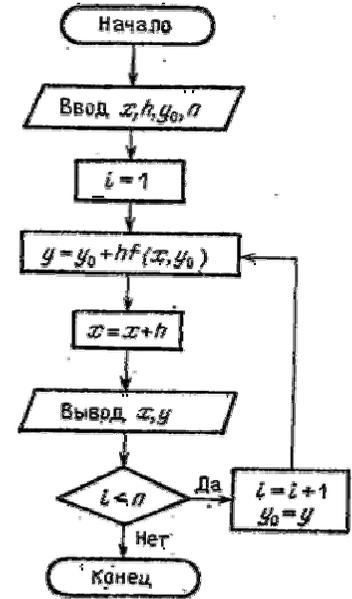
Формула Симпсона предназначена для вычисления определенного интеграла от функции $f(x)$. Значение интеграла представляется суммой

$$\int_a^b f(x) dx \approx \frac{h}{3} [y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n]$$

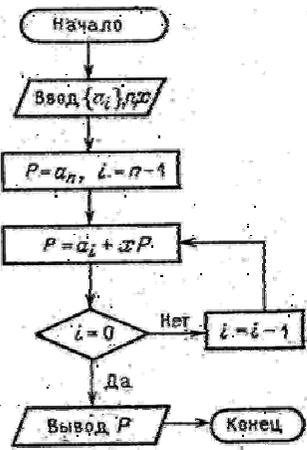


3. Блок-схема метода прогонки. Метод прогонки является частным случаем метода Гаусса для решения системы линейных уравнений с трехдиагональной матрицей

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & a_n & b_n \end{pmatrix} \vec{X} = \vec{D}$$



4. Блок-схема метода Эйлера. Метод Эйлера предназначен для решения задачи Коши для обыкновенного дифференциального уравнения $\frac{dy}{dx} = f(x, y), y(x_0) = y_0$. Если считать что узлы сетки x_i равноотстоящими, т.е. $\Delta x_i = x_{i+1} - x_i = h = const$, то значение сеточной функции y_{i+1} в любом узле x_{i+1} вычисляется по ее значению в предыдущем узле по формуле $y_{i+1} = y_i + hf(x_i, y_i)$.

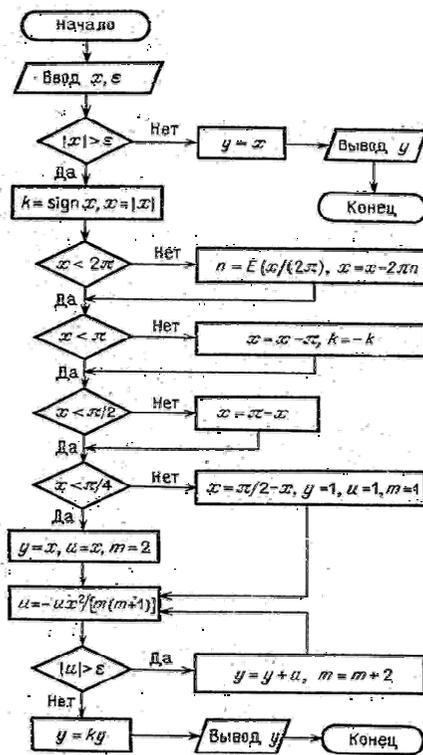


5. Блок-схема метода Горнера. При аппроксимации функций, а также в некоторых других задачах приходится вычислять значения многочленов вида

$$P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n.$$

Для исключения возведения x в степень в каждом члене многочлен в схеме Горнера переписывают в виде

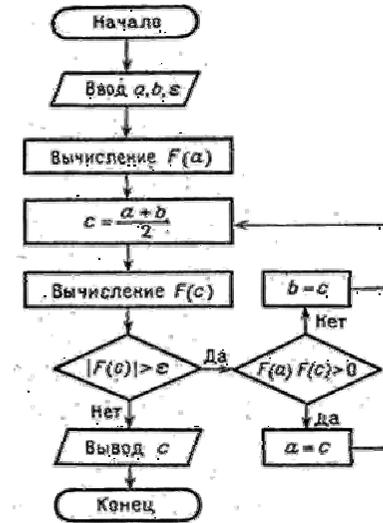
$$P(x) = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + xa_n)\dots)).$$



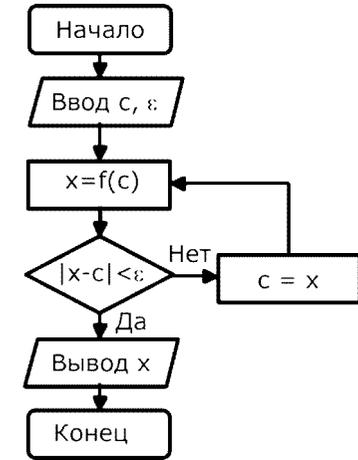
6. Блок-схема вычисления синусов. Значение синуса можно вычислить с помощью ряда

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

При вычислении данной суммы учитываются только члены, которые больше некоторого малого числа $\epsilon > 0$, характеризующего точность вычисления.



7. Блок-схема метода деления отрезка пополам. Это метод нахождения корней нелинейного уравнения $f(x) = 0$. Допустим, что удалось найти отрезок $[a, b]$, в котором расположено исходное значение корня $x = c$, т.е. $a < c < b$. В качестве начального приближения корня c_0 принимаем середину этого отрезка, т.е. $c_0 = (a+b)/2$. Далее исследуются значения функции на концах отрезков $[a, c_0]$ и $[c_0, b]$. Тот из них, на концах которого $f(x)$ принимает значения разных знаков, содержит искомый корень. Этот отрезок принимается в качестве исходного отрезка. Процесс повторяем итерационно, при этом отрезок, содержащий



8. Блок-схема метода простой итерации. Метод простой итерации предназначен для нахождения корней уравнения $F(x) = 0$. Для использования этого метода исходное нелинейное уравнение записывается в виде $x = f(x)$. Пусть известно начальное приближение корня $x = c_0$. Создаем итерационный процесс $c_{n+1} = f(c_n)$, $n = 1, 2, \dots$. Итерационный процесс прекращается, если результаты двух последовательных итераций близки: $|c_{n+1} - c_n| < \epsilon$. Достаточным условием сходимости метода простой итерации является условие $|f'(c_n)| < 1$.

корень сокращается после n -й итерации в 2^n раз.

14. ЛЕКЦИЯ № 14

DELPHI - СРЕДА ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ СОЗДАНИЕ ПРИЛОЖЕНИЙ В DELPHI. ОБЩАЯ ХАРАКТЕРИСТИКА ВИЗУАЛЬНЫХ КОМПОНЕНТОВ. ФОРМЫ И МЕНЮ.

14.1. Модули

Модули являются инструментом для разработки библиотек прикладных программ и являются основаниями для технологии программирования, которая называется модульным программированием.

Модуль - это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторую последовательность операторов.

В модулях явным образом выделяется некоторая "видимая" интерфейсная часть, в которой содержатся описания глобальных типов, констант и переменных, а также приводятся заголовки глобальных процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполнительной части модуля, которая может быть скрыта от пользователя.

Замечание. Важная особенность модулей заключается в том, что компилятор Turbo Pascal размещает их программный код в отдельном сегменте памяти. Максимальная длина сегмента памяти не может превышать 64 КБ, однако количество одновременно используемых модулей ограничивается лишь доступной памятью, что дает возможность создавать большие программы.

Модуль имеет следующий синтаксис:

```
unit <имя модуля> interface
<интерфейсная часть модуля> implementation
<исполняемая часть модуля> begin
<иницизирующая часть модуля> end.
```

Заголовок модуля состоит из зарезервированного слова *unit* и следующего за ним имени модуля. Это имя должно совпадать с именем дискового файла в который помещается исходный текст модуля. Если, к примеру, имеем заголовок модуля *unit global*, то исходный текст соответствующего модуля должен размещаться в дисковом файле *global.pas*.

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

```
uses <список модулей>;
```

где: *uses* - зарезервированное слово,

<Список модулей> - список модулей, с которыми устанавливается связь; элементами списка являются имена модулей, отделённые друг от друга запятыми, например:

```
uses crt,graph,global;
```

Если объявление *uses* используется, то оно должно открывать раздел описаний основной программы. В модуле можно использовать другие модули. Предложение *uses* в модулях может следовать либо сразу за словом *interface*, или сразу за словом *implementation*.

Интерфейсная часть модуля открывается словом *interface*. В этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных, процедур и функций), которые должны стать доступными основной программе и другим модулям. Отметим, что объявление процедур и функций в интерфейсной части автоматически сопровождается их компиляцией. Таким образом, обеспечивается доступ к процедурам и функциям из основной программы, а также других модулей.

Следует учесть, что все константы и переменные, объявленные в интерфейсной части модуля, равно как и глобальные константы и переменные основной программы, помещаются компилятором в общий сегмент данных. Порядок появления различных разделов объявления и их количество может быть произвольным.

Если в интерфейсной части объявляются внешние процедуры и функции, их тела должны следовать сразу за их заголовками в исполняемой части модуля.

Исполняемая часть модуля начинается словом *implementation* и содержит описания процедур и функций, объявленных в интерфейсной

части модуля. В ней могут объявляться локальные для модуля объекты - вспомогательные типы, константы, переменные и блоки.

Описанию подпрограмм, объявленной в интерфейсной части модуля, в исполняемой части должен предшествовать заголовок, в котором можно опускать список формальных переменных (и тип результата для функции), т.к. они уже описаны в интерфейсной части. Но если заголовок подпрограммы приводится в полном виде, он должен совпадать с заголовком, объявленным в интерфейсной части.

Локальные переменные и константы, а также все программные коды, порожденные при компиляции модуля, помещаются в общий сегмент памяти.

Иницилирующая часть завершает модуль. Она может отсутствовать вместе с начинающим ее словом *begin* или может быть пустой - тогда за *begin* сразу следует признак конца модуля (*end.*).

В иницилирующей 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 них могут иницироваться переменные, открываться нужные файлы и т.п.

Не рекомендуется делать иницилирующую часть пустой, лучше её опустить: пустая часть содержит пустой оператор, которому будет передано управление при запуске программы.

В среде Turbo Pascal имеются средства, управляющие способом компиляции модулей и облегчающие разработку крупных программных проектов. В частности, определены три режима компиляции:

compile, *make* и *build*. Режимы компиляции отличаются только способом связи компилируемого модуля или основной программы с другими модулями, объявленными в предложении *uses*.

При компиляции модуля или основной программы в режиме *compile* все упоминающиеся в предложении *uses* модули должны быть предварительно откомпилированы и результаты компиляции помещены в одноименные файлы с расширением *.tpu* (от англ. Turbo Pascal Unit).

Например, если в программе (модуле) имеется предложение *uses global;* то на диске в каталоге, объявленном опцией *unit directories*, уже должны

находиться файлы *global.tpu*. Файлы с расширением *.tpu* создаются в результате компиляции модуля.

В режиме *make* компилятор проверяет наличие *tpu*-файлов для каждого объявленного модуля. Если какой-либо из файлов не обнаружен, система пытается отыскать одноименный файл с расширением *.pas*, т.е. файл с исходным текстом модуля, и, если *pas*-файл найден, приступает к его компиляции. Кроме того, в этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если в *pas*-файл внесены какие-либо изменения, то независимо от того, есть ли уже в каталоге соответствующий *tpu*-файл или нет, система осуществляет его компиляцию перед компиляцией основной программы. Более того, если изменения внесены в интерфейсную часть модуля, то будут перекомпилированы также и другие модули, обращающиеся к нему. Режим *make*, таким образом, существенно облегчает процесс разработки крупных программ с множеством модулей: программист избавляется от необходимости следить за соответствием существующих *tpu*-файлов их исходному тексту, т.к. система делает это автоматически.

В режиме *build* существующие *tpu*-файлы игнорируются, и система пытается отыскать (и компилировать) соответствующий *pas*-файл для каждого объявленного в предложении *uses*-модуля. После компиляции в режиме *build* программист может быть уверен в том, что учтены все сделанные им изменения в любом из модулей. Подключение модулей к основной программе и их возможная компиляция осуществляется в порядке их объявления в предложении *uses*.

При переходе к очередному модулю система предварительно отыскивает все модули, на которые он ссылается. Ссылки модулей друг на друга могут образовывать древовидную структуру любой сложности, однако запрещается явное или косвенное обращение модуля к самому себе.

Turbo Pascal разрешает ссылки на частично откомпилированные модули, что приблизительно соответствует опережающему описанию подпрограммы. Если интерфейсные части любых двух модулей независимы, Turbo Pascal сможет идентифицировать все глобальные идентификаторы в каждом из модулей, после чего откомпилирует тела модулей обычным способом.

Понятие об "объектном типе данных".

Тип данных "объект" - это ключевое понятие технологии программирования, которая называется объектно-ориентированным программированием (ООП). Тип данных "объект" очень "похож" на тип данных "запись", но с одним существенным отличием: тип данных "объект" включает в себя не только данные, но также процедуры и функции, называемые методами. Методы оперируют данными объекта, т.е. они порождают, модифицируют и удаляют данные объекта. Это отличие является первым из трех свойств (инкапсуляция, наследование, полиморфизм), характеризующих объекты.

Инкапсуляция представляет включение процедур и функций в тип данных "запись" для получения нового типа данных - "объект".

Поведение объекта, заданное его методами, и свойства, определяемые данными различных типов, формируют абстракцию, которая называется объектом. Такое представление больше соответствует обычной человеческой логике и позволяет разрабатывать программы на логическом уровне.

Простейший пример описания объектного типа выглядит так:

```
type complex=object
re: real; im: real
end;
```

где для формирования структуры используется служебное слово Object. Для объектных типов так же, как и для записей, допускается использование оператора присоединения *with ... do ...*. Вызов метода объекта называют передачей сообщения объекту.

Замечание. Компилятор Turbo Pascal позволяет непосредственно обращаться к полям объекта, используя обычную точечную нотацию (*read(x.re,x.im)*). Однако это считается отступлением от объектно-ориентированного стиля, согласно которому все действия с информацией, заданной в объектном типе, осуществляются только посредством его методов.

Опосредованное обращение к данным позволяет избежать во многих случаях непредвиденных изменений параметров. С этой целью используется зарезервированное слово *private*, запрещающее непосредственное обращение к тем или иным данным и методам объекта вне модуля, в котором описан объект. В версии 7.0 приватная секция может

располагаться в любом месте объекта. Если приватная секция находится не в конце объекта, то для ограничения диапазона действия зарезервированного слова *private* следует после приватной секции поместить слово *public*.

14.2. Задание для контроля знаний

Определите, что будет напечатано на экране компьютера в ходе выполнения соответствующей программы.

Варианты задания 14.2.

- | | |
|---|--|
| 1) <i>var k,m,a,b:</i>
<i>integer;</i>
<i>begin k:=0;</i>
<i>for m:=1 to 10 do</i>
<i>begin</i>
<i>a:=trunc(m/4);</i>
<i>b:=(11-m) mod 3;</i>
<i>if a<>b then</i>
<i>k:=k+1;</i>
<i>end;</i>
<i>write(k)</i>
<i>end.</i> | 2) <i>var i,k: integer;</i>
<i>a: array[1..10] of</i>
<i>integer;</i>
<i>begin</i>
<i>k:=0;</i>
<i>for i:=1 to 10 do</i>
<i>a[i]:=i</i>
<i>for i:=1 to 10 do</i>
<i>if trunc(a[i]/3)>2</i>
<i>then k:=k+1;</i>
<i>write(k)</i>
<i>end.</i> |
| 3) <i>var k,m: integer;</i>
<i>begin</i>
<i>k:=0;</i>
<i>for m:=1 to 10 do</i>
<i>if m mod 3<>m mod 5</i>
<i>then k:=k+1;</i>
<i>write(k)</i>
<i>end.</i> | 4) <i>var i,k: integer;</i>
<i>a: array[1..10] of</i>
<i>integer;</i>
<i>begin</i>
<i>k:=0;</i>
<i>for i:=1 to 10 do</i>
<i>a[i]:=i;</i>
<i>for i:=1 to 10 do</i>
<i>if trunc(a[i]/2)<=2</i>
<i>then k:=k+1;</i>
<i>write(k)</i>
<i>end.</i> |
| 5) <i>var k,m,n: integer;</i> | 6) <i>var i,k: integer;</i> |

```

begin k:=0;
for m:=1 to 10 do
begin n:=m mod 4;
if n>=m mod 3
then k:=k+1
end;
write(k) end.

```

```

7) var k,m,a,b:
integer;
begin k:=0;
for m:=1 to 10 do
begin
a:=trunc(m/4);
b:=(11-m) mod 2;
if a=b then
k:=k+1
end;
write(k) end.

```

```

9) var k,m,a,b:
integer;
begin k:=0;
for m:=1 to 10 do
begin a:=m mod 3;
b:=(11-m) mod 2;
if a>b then
k:=k+1
end;

```

```

a: array[1..5] of
integer;
begin
k:=1;
for i:=1 to 5 do
a[i]:=i mod 2+i;
for i:=1 to 5 do
if a[a[i] mod 4+1]<=i
then k:=k*a[i];
write(k)
end.

```

```

8) var i,k,n,m:
integer;
a: array[1..10] of
integer;
begin k:=0;
for i:=1 to 10 do
a[i]:=i+3;
for i:=1 to 10 do
begin n:=a[i] mod 4
m:=(11-i) mod 3
if n>m then
k:=k+1
end; write(k)
end.

```

```

10) var i, k, n : integer;
a: array[1..10] of
integer;
begin
k:=0;
for i:=1 to 10 do
a[i]:=i;
for i:=1 to 10 do
begin n:=a[i] div 4;

```

```

write(k) end.

```

```

11) var k,m,a,b:
integer;
begin k:=0;
for m:=1 to 10 do
begin
a:=trunc(m/5);
b:=m mod 3;
if a=b then
k:=k+1
end;
write(k) end.

```

```

13) var k,m,a,b: integer;
begin k:=0;
for m:=1 to 12 do
begin
a:=m mod 3;
b:=m mod 5;
if a>b then
k:=k+1;
end;
write(k) end.

```

```

if n=a[i] mod 3 then
k:=k+i mod 3
end; write(k)
end.

```

```

12) var i,k,n : integer;
a: array[1..10] of
integer;
begin
k:=0;
for i:=1 to 10 do
a[i]:=i;
for i:=1 to 10 do
begin n:=a[i] div 4;
if n=(i+2) mod 3 then
k:=k+i mod 3
end; write(k)
end.

```

```

14) var i,k,n: integer;
a: array[1..10] of integer;
begin k:=0;
for i:=1 to 10 do
a[i]:=i+i mod 3;
for i:=1 to 10 do
begin
n:=a[i] mod 3;
if trunc(a[i]/4)=n
then
k:=k+i mod 3
end;
write(k) end.

```

15. ЛЕКЦИЯ № 15

ЯЗЫК ОБЪЕКТ PASCAL ОСНОВНЫЕ ПОНЯТИЯ. ЭЛЕМЕНТЫ ЯЗЫКА. ТИПЫ ДАННЫХ. ПРОСТЫЕ ТИПЫ ДАННЫХ. ПРОСТЫЕ И СТРУКТУРИРОВАННЫЕ ОПЕРАТОРЫ ЯЗЫКА. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ.

15.1. Графика

В языке Turbo Pascal имеется значительное количество графических процедур и функций. Нам понадобятся лишь некоторые из них. Для того, чтобы компилятор "узнавал" их названия, мы должны после заголовка программы разместить строчку следующего вида:

uses graph; (что в переводе на русский язык означает "используется графика").

До сих пор во время нашей работы за компьютером экран всегда находился в текстовом режиме, поэтому на экране можно было видеть только символы (правда, включая такие "изысканные", как так называемые "символы псевдографики").

Для рисования прямых, окружностей и пр. необходимо перевести экран в графический режим. Для включения графического режима используется процедура *initgraph*. Простейшая программа может иметь вид:

Пример 15.1.

program prim1;

```
uses graph; {подключение стандартного библиотечного модуля для  
формирования графических изображений}  
var gd, {графический драйвер}  
gm: integer; {графический режим}  
begin  
gd:=vga; {графический драйвер: vga}  
gm:=vgaHi; {графический режим : vgaHi (640x480)}  
initgraph(gd,gm,""); {включить графический режим. в апострофы  
помещается маршрут к файлу egavga.bgi }  
if graphresult=grok then {если графический режим включился успешно, то  
нарисовать отрезки прямых...}  
begin  
line(0,0,639,479); line(0,479,639,0);  
readln; closegraph
```

end

end.

Мы видим, что процедура *initgraph* имеет три параметра. В качестве первых двух фактических параметров должны располагаться имена целых (*integer*) переменных. Не будем вдаваться в подробности, почему это так; вместо этого выясним их предназначение.

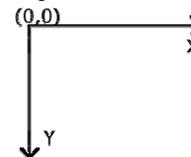
Первый параметр является кодом графического адаптера (т.е. электронной схемы, управляющей выводом информации на экран). (Дело в том, что на IBM-совместимых компьютерах применяется ряд стандартных графических адаптеров, носящих названия *CGA*, *EGA*, *VGA*.)

Каждый графический адаптер позволяет использовать несколько графических режимов, отличающихся количеством цветов и разрешающей способностью (в дальнейшем мы узнаем, что это такое). И второй из параметров как раз предназначен для того, чтобы указать, какой из графических режимов следует включить. Пока что мы ограничимся лишь одним графическим режимом *VGAHi*.

Третий параметр является строкой, содержащей путь к файлу, который называется *egavga.bgi*. В этом файле содержится драйвер (такая специальная программа), необходимый для работы с адаптерами *EGA* и *VGA*. И, как легко увидеть из нашего примера, файл этот находится в подкаталоге *TP\BGI*.

Все вышеизложенное необходимо знать каждому грамотному пользователю IBM-совместимых компьютеров. Однако в лабораторной работе достаточно использовать конструкцию, использованную в первом примере, для включения графического режима. (И не страшно, если в ней не все понятно.)

Для того, чтобы мы могли что-либо нарисовать на экране, нам нужно уметь задавать положение на экране того, что мы рисуем. Для этого с экраном связывается система координат следующего вида:



Каждая точка на экране на самом деле представляет собой очень маленький прямоугольник (и поскольку это не совсем точка, то иногда используют специальный термин - "пиксел"). Количество точек (пикселов), умещающихся на экране по вертикали и горизонтали, называют разрешающей способностью. Разрешающая способность экрана в режиме VGAHi - 640x480. Это означает, что по горизонтали на экране умещается 640 точек, а по вертикали - 480.

Точка в левом верхнем углу экрана имеет координаты (0,0). Координата X любой точки экрана лежит в пределах от 0 до 639, а координата Y - в пределах от 0 до 479.

Процедура *line(x1,y1,x2,y2)* рисует на экране прямую, соединяющую точки (x1,y1) и (x2,y2).

Пример 15.2. Изобразить на экране прямоугольный треугольник с вершинами (320,10), (120,210), (520,210).

```
program prim2; {изображение прямоугольного треугольника }
uses graph;
var gd,gm: integer;
begin
gd:=vga;
gm:=vghi;
initgraph (gd,gm,"");
if graphresult=grok then
begin
line(120,210,520,210); {горизонтальный отрезок}
line(120,210,320,10); {левый катет треугольника}
line(320,10,520,210); {правый катет треугольника}
readln;
closegraph
end
end.
```

Для задания цвета рисования прямых, окружностей, точек и пр. используется процедура *setcolor*. В качестве единственного ее параметра нужно указать целое число, являющееся кодом цвета. Цвета кодируются в соответствии со следующей кодовой таблицей:

<i>black</i> =0 - черный	<i>lightred</i> =12 - розовый
<i>darkgray</i> = 8 - темно-серый	<i>red</i> =4 - красный
<i>blue</i> =1 - синий	<i>lightmagenta</i> =13 - светло-сиреневый
<i>lightblue</i> = 9 - голубой	<i>magenta</i> =5 - сиреневый
<i>green</i> =2 - зеленый	<i>yellow</i> =14 - желтый
<i>lightgreen</i> =10 - светло-зеленый	<i>brown</i> =6 - коричневый
<i>cyan</i> =3 - циановый	<i>white</i> =15 - белый
<i>lightcyan</i> =11 - светло-циановый	<i>lightgray</i> =7 - светло-серый
(цвет морской волны)	

Для закраски замкнутой области используется процедура *floodfill*, три целочисленных параметра которой задают начальную точку закраски и код цвета линии, ограничивающей область.

Цвет, которым будет производиться закраска, ничего общего не имеет с цветом, задаваемым процедурой *setcolor*. Цвет закраски задается вторым параметром процедуры *setfillstyle*. Первый параметр этой процедуры (задающий узор для закраски) на первых порах будем задавать равным единице (что соответствует сплошной закраске).

Рисование прямоугольников - часто встречающаяся проблема, и поэтому неудивительно, что существует стандартная процедура *rectangle*. Так же, как и для процедуры *rectangle*, мы должны указать четыре числа - координаты двух противоположных углов прямоугольника. (Для процедуры *bar* цвет задается не с помощью *setcolor*, а с помощью *setfillstyle*!).

При рисовании сложных изображений, содержащих много отрезков, возникает довольно противная проблема - вычислять координаты всех точек. Если использовать процедуру *linereel*, то достаточно указывать смещения по обеим координатам относительно текущей точки. Для относительного перемещения без рисования используется процедура *moverel*. Для задания начальных значений координат текущей точки используется процедура *moveto*.

Если Вас интересуют другие графические процедуры или функции, то Вам следует обратиться к системе "помощи" (*help*).

Движение реализуется с помощью процедур *getimage* и *putimage*. Процедура *getimage* запоминает образ изображаемого объекта и образ области экрана такого же размера, закрашенной цветом фона. Процедура

putimage на каждом шаге последовательно заменяет старое изображение цветом фона и создает изображение на новом месте.

Пример 15.3. Управление движением объекта.

Направление движения определяется нажатой клавишей (стрелки влево, вправо, вверх, вниз). Шаг перемещения вводится. Реализация движения характеризуется тем, что на каждом шаге запоминается образ области экрана, куда помещается курсор, затем при смещении курсора изображение восстанавливается.

program prim3;

```
{Программа управления движением курсора.}
{Курсор - прямоугольный объект, двигающийся вверх, вниз,}
{Вправо, влево при нажатии соответствующих стрелок.}
{Выход при нажатии клавиши esc (код 27)}
uses crt,graph; {модуль crt необходим для использования функции readkey}
var p, {указатель на образ изображения "под" курсором}
pc: pointer; {указатель на образ курсора}
grm,grd, {переменные для номеров графических драйверов и режима}
curx,cury, {координаты текущего положения курсора}
curx0,cury0, {переменные для запоминания координат курсора}
lx,ly, {ширина и длина курсора прямоугольного вида}
hx,hy : integer; {шаги движения курсора по горизонтали и вертикали}
size, {количество байтов для изображения курсора}
c: word; {цвет изображения}
ch : char;
procedure badkey; {процедура формирует звук при нажатии
"неправильной" клавиши }
begin
sound(500); delay(100);
sound(400);delay(200);
nosound
end;
begin
write('Введите размеры курсора и шаги движения: ');
read(lx,ly,hx,hy); {установка значения системной переменной для
обеспечения совместимости работы модулей crt и graph}
```

```
directvideo:=false; grd:=0; initgraph(grd,grm,"); {инициализация
графического режима с автоматическим определением подходящего
драйвера}
```

```
c:=getcolor; size:=imagesize(0,0,lx,ly);
getmem(pc,size); getmem(p,size); {выделяются области для хранения образа
курсора, и образа изображения под курсором}
setfillstyle(1,c); {устанавливается тип и цвет закрашки курсора}
getimage(0,0,lx,ly,p^); {p указывает на область памяти, где хранится
изображение, которое будет "закрето" курсором}
curx:=0; cury:=0; bar(0,0,lx,ly);
getimage(0,0,lx,ly,pc^); {pc указывает на область памяти с изображением
курсора}
setcolor(6); setfillstyle(1,2);
bar3d(150,150,200,30,10,true); {параллелепипед, на фоне которого
происходит движение}
repeat {цикл по вводу символа}
ch:=readkey; {ввод очередного символа}
if ord(ch)=0 then {нажата управляющая клавиша}
begin
ch:=readkey;
curx0:=curx; cury0:=cury;
case ord(ch) of
77: if curx<getmaxx-hx then curx:=curx+hx; {шаг вправо}
75: if curx>hx then curx:=curx-hx; {шаг влево}
72: if cury>hy then cury:=cury-hy; {шаг вверх}
80: if cury<getmaxy-hy then cury:=cury+hy {шаг вниз} else badkey
end;
if (curx<>curx0) or (cury<>cury0) then
begin
putimage(curx0,cury0,p^,0); {восстановить изображение, которое было
"закрето" курсором}
getimage(curx,cury,curx+lx,cury+ly,p^); {запомнить то изображение,
которое будет "закрето" курсором}
putimage(curx,cury,pc^,0); {установить курсор в новое положение}
end
```

```

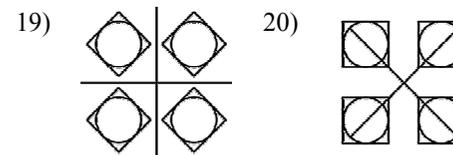
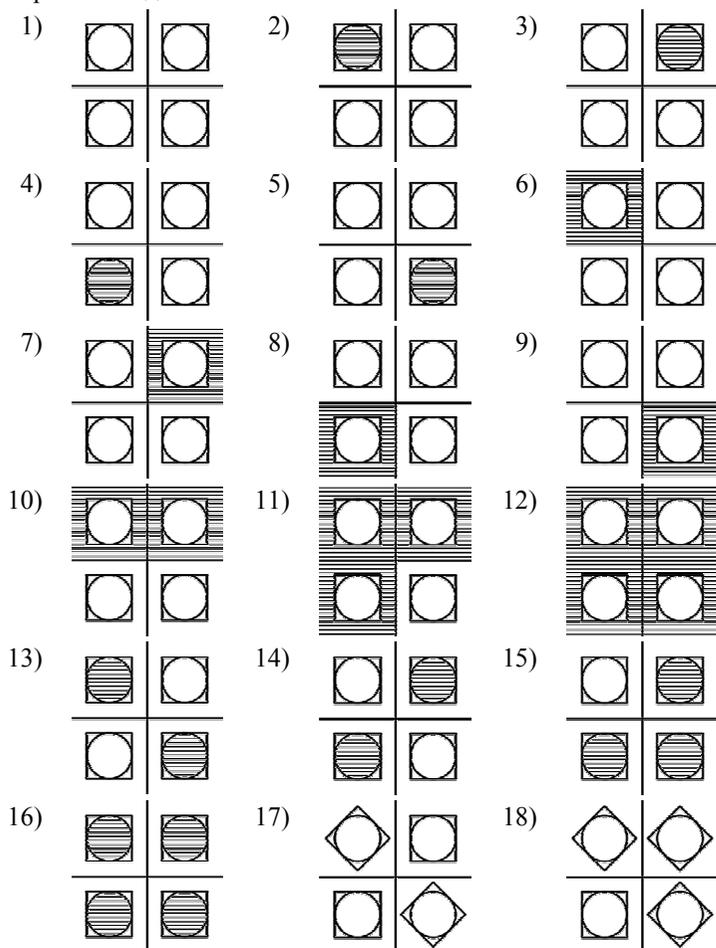
end
else badkey until ord(ch)=27;
closegraph
end.

```

15.2. Задание для контроля знаний

Написать программу, которая на экране компьютера рисует соответствующую картинку.

Варианты задания 15.2.



СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Малые жанры русского фольклора. Хрестоматия: Учебн. пособие для филол. спец. вузов/Сост. В.Н. Морохин.– 2-е изд., испр. и доп.– М.: Высш. Шк., 1986.– 399 с., стр. 73
2. Гурьянов А.Е. Алгол-60 в упражнениях. Учебное пособие.– Л.: изд-во Ленингр. Ун-та, 1978.– 160 с.
3. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0.– М.: Диалог-МИФИ, 1997.– 288 с.
4. Трифонов Н.П., Пасхин Е.Н. Практикум работы на ЭВМ.– М.: Наука, Главная редакция физико-математической литературы, 1982.– 288 с.
5. Немошкаленко В.В., Антонов В.Н. Методы вычислительной физики в теории твердого тела. Зонная теория металлов.– Киев: Наук. Думка, 1985.– 408 с.
6. Варшалович Д.А., Москалев А.Н., Херсонский В.К. Квантовая теория углового момента.– Л.: Наука, Ленингр. отд., 1975.– 439 с.
7. Світ і людина: Довідник.– Х.: Прапор, 1998.– 576 с.
8. Малая энциклопедия современных знаний/ составитель Менделеев В.А.– Х.: Торгсинг, 1998.– 768 с.
9. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию.– М.: Наука. Гл. ред. физ.-мат. лит., 1988.– 224 с.
10. Вирт Н. Алгоритмы и структуры данных: Пер. с англ.– СПб.: Невский Диалект, 2001.– 352 с.
11. Грогано П. Программирование на языке Паскаль: Пер. с англ.– М.: Мир, 1982.– 384 с.
12. Гуденко Д.А., Петrochenko Д.В. Сборник задач по программированию.– СПб.: Питер, 2003.– 475 с.
13. Дал У., Дейкстра Э., Хоор К. Структурное программирование: Пер. с англ.– М.: Мир, 1975.– 248 с.
14. Долинский М.С. Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач : Учебное пособие.– СПб.: Питер, 2005.– 237 с.
15. Йодан Э. Структурное проектирование и конструирование программ.– М., Мир, 1979.– 416 с.

16. Немнюгин С.А. Turbo Pascal: практикум.– СПб.: Питер, 2003.– 256 с.
(Серия «Учебники для вузов»)
17. Окулов С.М. Основы программирования.– М.: ЮНИМЕДИАСТАЙЛ,
2002.– 424 с.

НАЗНАЧЕНИЕ ФУНКЦИОНАЛЬНЫХ КЛАВИШ СРЕДЫ TURBO PASCAL

Функциональные клавиши обозначаются F1, F2,..., F12 и располагаются в самом верхнем ряду клавиатуры. Приведем команды, которые передаются функциональными клавишами и некоторыми их комбинациями с клавишами *Ctrl* и *Alt*:

F1 – обратиться за справкой к встроенной справочной службе (*Help* – помощь);

F2 – записать редактируемый текст в дисковый файл;

F3 – прочитать текст из дискового файла в окно редактора;

F4 – используется в отладочном режиме: начать или продолжить исполнение программы и остановиться перед исполнением той ее строки, на которой стоит курсор;

F5 – распахнуть активное окно на весь экран;

F6 – сделать активным следующее окно;

F7 – используется в отладочном режиме: выполнить следующую строку программы; если в строке есть обращение к процедуре (функции), войти в эту процедуру и остановиться перед исполнением первого ее оператора;

F8 – используется в отладочном режиме: выполнить следующую строку программы; если в строке есть обращение к процедуре (функции), исполнить ее и не проследивать ее работу;

F9 – компилировать программу, но не выполнять ее;

F10 – перейти к диалоговому выбору режима работы с помощью главного меню;

Ctrl-F9 – выполнить прогон программы: компилировать программу, находящуюся в редакторе, загрузить ее в оперативную память и выполнить, после чего вернуться в среду Turbo Pascal;

Alt-F5 – сменить окно редактора на окно вывода результатов работы(прогона) программы;

Alt-X – выход из Turbo Pascal.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять по курсу

“ Основи інформаційних технологій та програмування”
для студентів фізико-технічних спеціальностей

Російською мовою

Укладачі: САВЧЕНКО Микола Володимирович

КУХАРЕНКО Володимир Миколайович

Відповідальний за випуск В.М. Свистунов

Роботу до видання рекомендував О.П. Сук

За авторською редакцією

Зав. редакційно-видавничим

відділом М.П. Єфремова

План 2006 р., п. 10/176-05

Підп. до друку 26.12.2005 р. Формат 60x84 1/16. Папір офсетний. Riso-друк.

Гарнітура Таймс. Ум. друк. арк. 3,5. Обл.-вид. арк.4,4.

Наклад 50 прим. Зам. № 407. Ціна договірна.

Видавничий центр НТУ “ХП”.

Свідоцтво про державну реєстрацію ДК №116 від 10.07.2000 р.

61002, Харків, вул. Фрунзе, 21.

Друкарня НТУ “ХП”. 61002, Харків, вул. Фрунзе, 21.