1. ЧИСЛОВЫЕ ТИПЫ. ОПЕРАТОР ПРИСВАИВАНИЯ

1.1*. Указать неправильные записи идентификаторов:

а) task1.1 б) task1_1 в) task1v1 г) GoTo
д) 4you е) x_1 ж) LXIV з) π

1.2. Ответить на следующие вопросы:

а) Верно ли, что в идентификаторах языка Паскаль большие и малые одноименные буквы не различаются? Например, *N* и *n* – это один и тот же идентификатор или нет?

б) В чем отличие служебных слов от имен (других идентификаторов)? Можно ли, например, использовать служебное слово *mod* для обозначения переменных программы?

в) Чем отличаются стандартные имена от других имен? Можно ли, например, использовать стандартное имя *integer* для обозначения переменных программы?

г) Что общего между служебными словами и стандартными именами и в чем различие между ними?

1.3. Ответить на следующие вопросы:

а) Почему в языке Паскаль знак умножения всегда выписывают явно (например, записывают *a*t*, а не *at*)?

б) Почему аргумент функции всегда записывают в скобках (например, записывают *ln*(5), а не *ln*5)?

1.4*. Записать по правилам языка Паскаль следующие целые числа:

а) 5!; б) LXIV; в) -10^4; г) 000023

1.5*. Вычислить результаты следующих операций:

а) 20 div 6; б) 20 mod 6; в) 20 div 4;
г) 20 mod 4; д) (–20) div 6; е) (–20) mod 6;
ж) (–20) div 4; з) (–20) mod 4; и) 2 div 5;
к) 2 mod 5; л) 123 div 0; м) 2 mod (–3)

1.6*. Вычислить значения следующих стандартных функций:

а) abs(3); б) abs(–3); в) sqr(3);
г) sqr(–3); д) succ(3); е) succ(–3);
ж) pred(3); з) pred(–3); и) ord(3);
к) ord(–3); л) ord(0); м) succ(ord(pred(3))

1.7. Почему в Паскале факториал от 10 нельзя записать в виде 10! или 1*2*3*...*10 ?

1.8. Что в Паскале обозначает стандартное имя *maxint*? Можно ли записать *maxint*+1?

1.9*. В языке Паскаль запись вещественного числа (например, +35.478E–2) в общем случае состоит из трех частей: целой части со знаком (+35), дробной части с точкой (.478) и десятичного порядка (E–2). Какие из этих частей можно опускать при записи вещественного числа? Определить, какие из следующих записей вещественных чисел неправильны:

а) .478E–2; б) +35E–2; в) +35.478;
г) +35; д) .478; е) E–2

1.10*. Одинаковы ли записи 12E–4 и 12e–4 ?

1.11. Есть ли разница (с точки зрения языка Паскаль) между числами 100 и 100.0, между 20 и 2E1? По какому признаку (величине или форме записи) целые числа отличаются от вещественных? Как именно отличить целое число от вещественного?

1.12. Верно ли, что в общем случае вещественные числа представляются в компьютере приближенно и операции над ними выполняются неточно? Можно ли утверждать, что всегда выполняется равенство (1.0/3.0)*3.0 = 3.0?

Ответить на эти же вопросы относительно целых чисел и равенства (4 div 2)*2 = 4.

1.13*. Записать по правилам языка Паскаль следующие числа как вещественные, указывая в дробной части до 4 цифр:

а) 10/2; б) $\sqrt{4}$; в) 6,38; г) –0,7(4);
д) 11/4; е) –1/6; ж) $\sqrt{2}$; з) π;
и) $5 \cdot 10^6$; к) $-24{,}8 \cdot 10^{-7}$; л) 10^6; м) 1/100000

1.14*. Записать следующие вещественные числа без десятичного порядка:

а) –0.00027E+4; б) 666E–3; в) 1E1; г) 1E–1

1.15*. Указать неправильные записи чисел и объяснить причину ошибки:

а) 0006; б) –0; в) 7,0; г) 7.;
д) +0.3; е) .3; ж) 2/3; з) E–1;
и) 8E0; к) 0E–4; л) 2*E5; м) *e*

1.16. Верно ли, что везде, где по правилам языка Паскаль должно указываться вещественное число (выражение), можно указывать и целое число (выражение) и что при этом целое число автоматически заменяется на равное ему вещественное число?

Верно ли обратное утверждение - что вместо целых чисел (выражений) можно указывать вещественные числа (выражения)?

1.17*. Какие из следующих выражений правильные, а какие – нет? Для правильных выражений указать их типы (целый или вещественный) и значения.

а) 0+1;	б) 0.0+1;	в) 1–1E+1;	г) 5*1.0;
д) 4.0/2.0;	е) 4.0/2;	ж) 4/2.0;	з) 4/2;
и) 6 div 3;	к) 6.0 div 3;	л) 8 mod 8;	м) 5.2 mod 5.2

1.18. Указать типы и значения следующих стандартных функций:

а)* abs(–2);	б)* abs(–2.0);	в)* sqr(–5);	г)* sqr(–5.0);
д)* sqrt(16.0);	е)* sqrt(16);	ж)* exp(0.0);	з)* exp(0);
и) ln(1.0);	к) ln(1);	л) sin(0.0);	м) sin(0);
н) cos(0.0);	о) cos(0);	п) arctan(0.0);	р) arctan(0)

1.19*. Вычислить значения выражений:

а) trunc(6.9);	б) round(6.9);	в) trunc(6.2);	г) round(6.2);
д) trunc(–1.8);	е) round(–1.8);	ж) trunc(2);	з) round(–2);
и) trunc(0.5);	к) round(0.5);	л) trunc(–0.5);	м) round(–0.5)

1.20. Выразить функцию *round* через функцию *trunc*.

1.21. Выразить операцию *div* через функцию *trunc*.

1.22. Почему в языке Паскаль при записи формул их «вытягивают» в линию (например, пишут *x*/*y*, а не $\frac{x}{y}$, пишут *B1*, а не B_1)?

1.23. Верны ли следующие утверждения относительно записи пробелов в текстах Паскаль-программ? (Ниже знак _ явно обозначает пробел.)

а) Пробел нельзя указывать внутри чисел, но можно указывать внутри идентификаторов. Например, недопустимо число 12_34, но допустим идентификатор s_i_n

б) Между соседними идентификаторами и/или числами ставить пробел необязательно. Например, операцию деления нацело можно записать как 5_div_2 и как 5div2

в) В остальных местах пробел можно указывать, а можно и не указывать. Например, допустимы следующие записи:

x+2*z x_+2*_z sin(x) sin_(_x_) a_ : _=_5

г) Везде вместо одного пробела можно указывать любое число пробелов. Например, допустима следующая запись:

15_ _mod_ _ _6

1.24. Верны ли следующие утверждения относительно переноса текста на новую строку?

а) Переход на новую строку эквивалентен указанию пробела в этом месте текста программы, поэтому переходить на новую строку можно только там, где разрешено указывать пробел.

б) При переходе на новую строку надо обязательно повторить знак последней операции. Например, переносить на новую строку текст *x*+*y***z*/*x*–7 между * и *z* надо так:

x+y*
*z/x–7

1.25*. Убрать из выражения лишние скобки, т.е. такие, удаление которых не изменит порядок выполнения операций в выражении:

а) ((a*b) div c) mod (a+b);
б) (–n) mod m;
в) (sin(x+y))/2–cos((x+y)/2);
г) (a*x)/(b*y)+(a/x)/(b/y)

1.26. Указать порядок выполнения операций, тип и значение выражения:

а)* 24/(3*4)–24/3/4+24/3*4;
б) 3*7 div 2 mod 7/3 – trunc(sin(1));
в)* –5 mod 3 + (–5) mod 3;
г) succ(round(5/2)–trunc(5/2))+pred(ord(5)–5)*exp(0)

1.27*. Сколько операций выполняется при вычислении выражения (x+1/2)*(y+7/10)–3/4 ?
Как сократить число операций?

1.28*. Записать на Паскале следующие формулы:

а) $a+bx+cyz$;	б) $[(ax-b)\,x+c]\,x-d$;	в) $\frac{ab}{c}+\frac{c}{ab}$;
г) $\frac{x+y}{a_1}\cdot\frac{a_2}{x-y}$;	д) $(1+\frac{x}{2!}+\frac{y}{3!})/(1+\frac{2}{3+xy})$	е) $\lvert a+bx\rvert$;
ж) $10^4\alpha-3\frac{1}{5}\beta$;	з) $(1+x)^2$;	и) $\sqrt{1+x^2}$;
к) $ln(e^x+e^{-x})$;	л) $sin\ 8$;	м) $cos^2\,x^3$

1.29. Записать в общепринятой форме:

а)* (p+q)/(r+s)–p*q/(r*s);

б)* 1E3+beta/(x2–gamma*delta);

в) (–b+sqrt(sqr(b)–4*a*c))/(2*a);

г) a/b*(c+d)–(a–b)/b/c+1E–8;

д) x1+arctan(y2–alpha)/2*abs(x4–ln(5)*y5)/exp(–1)

1.30. Что следует делать, если по алгоритму нужна функция (например, тангенс) или операция (например, возведение в степень), которой нет в языке Паскаль?

Записать на Паскале следующие формулы:

а) $\operatorname{tg} x$;	б)* $\log_2 \frac{x}{5}$;	в) $\operatorname{ch} x$;	г) $\operatorname{arcctg} 10^3$;
д)* $\arcsin x$;	е)* x^{-1};	ж)* x^4;	з) x^{-2};
и) x^5;	к)* x^{100};	л) 2^x;	м) $x^{\sqrt{2}}$;
н)* $\sqrt[3]{1+x}$;	о) x^x	п)* max(x,y);	р) min(x,y)

1.31*. Как записать на Паскале величину основания натуральных логарифмов (число *e*), если Вы забыли цифры этого числа? А как записать число π в подобной ситуации?

1.32*. Как на Паскале записать синус от *x* градусов?

1.33. Записать на Паскале следующие формулы:

а) $\sqrt[8]{x^8+8^x}$; б) $\dfrac{xyz-3{,}3\,|x+\sqrt[4]{y}\,|}{10^7+\sqrt{\lg 4!}}$; в) $\dfrac{\beta+\sin^2\pi^4}{\cos 2+|\operatorname{ctg}\ \gamma|}$

1.34*. Пусть *N*=4321. Вычислить значения следующих выражений:

а) N mod 10; б) N div 10; в) N div 10 mod 10; г) N mod 100 div 10

1.35. Сформулировать правила выполнения оператора присваивания *v*:=*e*, где *v* – переменная, а *e* – выражение. Не является ли ошибкой запись *x*:=*x*+1, поскольку никакое число не может равняться самому себе плюс единица?

1.36. Записать оператор присваивания, который меняет знак у значения переменной *t*.

1.37*. Определить значение переменной *N* после выполнения следующих операторов:

N:=10; n:=2; N:=N+n

1.38*. Требуется поменять местами значения переменных *x* и *y*. Какой из следующих вариантов правильно решает эту задачу?

а) x:=y; y:=x; б) r:=x; x:=y; y:=r

1.39. Поменять местами значения переменных *x*, *y* и *z* так, чтобы в *x* оказалось значение переменной *y*, в *y* – значение переменной *z*, а в *z* – прежнее значение переменной *x*.

1.40. Записать на Паскале соответствующие операторы присваивания:

а)* $y = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$;

б) $f = 6{,}673 \cdot 10^{-8} \cdot \frac{m_1 \cdot m_2}{r^2}$;

в) $b = e^{|x-y|} + ln(1+e) \cdot log_2 tg\ 2$

1.41. Записать операторы присваивания, которые переменной *d* присваивают:

а) среднее арифметическое чисел *x*, *y* и *z*;

б) среднее геометрическое положительных чисел *x*, *y* и *z*;

в)* расстояние между точками с координатами (x_1, y_1) и (x_2, y_2);

г) корень уравнения $arctg(1+ln\ x) = \sqrt{2}$;

д)* площадь треугольника со сторонами *a*, *b* и *c*;

е)* дробную часть положительного числа *x*

1.42. В языке Паскаль значением вещественной переменной (скажем, *x*) может быть только вещественное число и в то же время допускается оператор присваивания, который вещественной переменной присваивает целое число (например, *x*:=7). Как в языке устраняется это противоречие?

1.43. Согласно правилу языка Паскаль, везде вместо вещественного выражения можно указывать целое выражение. А верно ли это утверждение, если в нем заменить слово «выражение» на слово «переменная»? Ответ обосновать.

1.44*. Если *y* – вещественная переменная, а *n* – целая, то какие из следующих операторов присваивания правильные, а какие нет и почему?

а) y:=n+1; б) n:=y–1; в) n:=4.0; г) y:=trunc(y);

д) n:=n div 2; е) y:=y div 2; ж) n:=n/2; з) n:=sqr(sqrt(n))

1.45. Правильны ли следующие операторы присваивания? Ответ обосновать.

а)* k:=k mod 3+k*cos(0);

б) x:=x*2 div 6+x/4

В упражнениях 1.46–1.54 требуется выписать последовательность операторов присваивания, решающих указанную задачу

1.46*. Присвоить целой переменной h третью от конца цифру в записи положительного целого числа k (например, если k=130985, то h =9).

1.47. Присвоить целой переменной d первую цифру из дробной части положительного вещественного числа x (так, если x=32.597, то d=5).

1.48. Целой переменной s присвоить сумму цифр трехзначного целого числа k.

1.49. Идет k-я секунда суток. Определить, сколько полных часов (h) и полных минут (m) прошло к этому моменту (например, h=3 и m=40, если k=13257= 3*3600+40*60+57).

1.50. Определить f – угол (в градусах) между положением часовой стрелки в начале суток и ее положением в h часов, m минут и s секунд ($0 \le h \le 11$, $0 \le m, s \le 59$).

1.51. Определить h – полное количество часов и m – полное количество минут, прошедших от начала суток до того момента (в первой половине дня), когда часовая стрелка повернулась на f градусов ($0 \le f < 360$, где f – вещественное число).

1.52. Пусть k – целое от 1 до 365. Присвоить целой переменной n значение 1, 2, ..., 6 или 7 в зависимости от того, на какой день недели (понедельник, вторник, ..., субботу или воскресенье) приходится k-й день невисокосного года, в котором 1 января – понедельник.

1.53. Материальная точка равномерно движется по окружности радиуса R с центром в начале координат и делает полный оборот за время $Tоб$. Вычислить координаты x и y этой точки в момент времени t, если в начальный момент она имела координаты (R,0).

1.54. Поменять местами значения целых переменных x и y, не используя дополнительные переменные.

2. ЛОГИЧЕСКИЙ ТИП

2.1. Ответить на следующие вопросы.

а) Какие величины входят в логический тип? Как они обозначаются?

б) Являются ли имена *true* и *false* стандартными? в) Можно ли внутри знака отношения (например, <=) указывать пробел?

г) Верно ли, что отношение – это выражение, принимающее значение *true* или *false*?

д) Верно ли, что в отношениях арифметические операции выполняются до операций отношения?

е) Можно ли в отношениях сравнивать целые числа с вещественными (например, 3>2.7)? А сравнивать числа с логическими величинами (например, *false*<>1)?

ж) Верно ли, что *false*<*true*?

2.2*. Вычислить значения отношений:

а) abs(x)>=1	при x= –2.4;
б) sqr(x)+sqr(y)<=4	при x=0.3, y= –1.6;
в) trunc(x)<round(x)	при x= –2.5;
г) k mod 7 = k div 5 – 1	при k=15;
д) a<>true	при a=true

2.3*. Записать на Паскале отношение, истинное при выполнении указанного условия и ложное в противном случае:

а) целое k делится на 7;

б) уравнение $ax^2+bx+c=0$ ($a \neq 0$) не имеет вещественных корней;

в) точка (x, y) лежит вне круга радиуса r с центром в точке (1, 0);

г) точка x принадлежит отрезку [–1, 1];

д) логические переменные a и b имеют разные значения;

е) натуральное число n является полным квадратом.

2.4*. Верно ли тождество $odd(N) \equiv (N \bmod 2=1)$, где N – любое целое?

2.5*. Вычислить выражения:

а) odd(5);	б) odd(4);	в) odd(0);
г) odd(–1);	д) succ(false);	е) succ(true);
ж) pred(false);	з) pred(true);	и) ord(false);
к) ord(true);	л) false<true;	м) ord(false)<ord(true);
н) odd(3)<odd(4);	о) pred(ord(false));	
п) ord(pred(false));	р) succ(false)<>succ(true)	

2.6*. Вычислить результаты логических операций:

а) not true;	б) not false;	в) not not false;
г) true and true;	д) true and false;	е) false and false;
ж) true or true;	з) true or false;	и) false or false

2.7. Упростить следующие выражения (*a* – произвольная логическая величина):

а)* *a*=true;	б)* not (*a*=true);	в)* not (*a*<>true);
г) not not *a*;	д) *a* and false;	е) *a* and true;
ж) *a* and *a*;	з) *a* and (not *a*);	и) *a* or false;
к) *a* or true;	л) *a* or *a*;	м) *a* or (not *a*)

2.8. Вычислить выражения:

а) not odd(n)	при n=0;
б) t and (p mod 3=0)	при t=true, p=101010;
в) (x*y<>0) and (y>x)	при x=2, y=1;
г) (x*y<>0) or (y>x)	при x=2, y=1;
д) a or (b>a)	при a=false, b=true;
е) odd(k) or odd(n)	при k=2, n= –4

2.9. Ответить на следующие вопросы.

а) Обязательно ли операндами логических операций должны быть логические величины?

б) Верно ли, что логические операции выполняются до операций отношения?

в)* Если отношение является операндом другой операции, то обязательно ли его надо заключать в круглые скобки?

г)* Какие из следующих записей ошибочны и почему?

1 and 2; not 2=5; 1>0 and 3<1; false<true = true

2.10. Записать логическое выражение, истинное при выполнении указанного условия и ложное в противном случае:

а)* 0<*x*<1;

б)* *x*=max(*x*, *y*, *z*);

в) *x*≠max(*x*, *y*, *z*) (операцию *not* не использовать);

г)* хотя бы одна из логических переменных *a* и *b* имеет значение *true*;

д)* обе логические переменные *a* и *b* имеют значение *true*;

е) целое *y* делится на 4, но не делится на 100;

ж) целое *n* делится на 7, а целое *k* делится на 13;

з) целое *n* делится на 2 или 5.

2.11. Указать порядок выполнения операций и вычислить значение выражения:

а) a or b and not a при a=true, b=false;

б) (a or b) and not a при a=true, b=false;

в)* a and b or not c and a при a=true, b=false, c=false;

г)* a and b > a or b при a=false, b=true;

д) (x>=0) or (y<0) and odd(x) or (y*y<>4) при x= –1, y=3

2.12. Записать логическое выражение, истинное при выполнении указанного условия и ложное в противном случае:

а) *x* принадлежит отрезку [0, 1];

б) *x* лежит вне отрезка [0, 1];

в)* *x* принадлежит отрезку [2, 5] или [–1, 1] (операцию *not* не использовать);

г)* *x* лежит вне отрезков [2, 5] и [–1, 1];

д) каждое из чисел *x, y, z* положительно;

е) хотя бы одно из чисел *x, y* и *z* положительно;

ж) ни одно из чисел *x, y* и *z* не является положительным;

з) только одно из чисел *x, y* и *z* положительно;

и) логическая переменная *a* имеет значение *true,* а логическая переменная *b* имеет значение *false*;

к)* год с порядковым номером *y* является високосным (год високосный, если его номер кратен 4, однако из кратных 100 високосными являются лишь кратные 400; например, 1700, 1800, 1900 и 2001 – невисокосные годы, а 1600 и 2000 – високосные).

2.13*. Верно ли, что при выполнении любой операции сначала вычисляются все ее операнды и только затем выполняется сама операция? Вычислить выражения:

а) true or (1/0>0); б) (1/0>0) or true

2.14. Нарисовать на плоскости (*x, y*) область, в которой и только в которой истинно указанное логическое выражение:

а)* (y>=x) and (y+x>=0) and (y<=1);

б) (sqr(x)+sqr(y)<1) or (y>0) and (abs(x) <=1);

в) (trunc(y)=0) and (round(x)=0);

г) (round(x)=trunc(x)) and (x*x+y*y<=4);

д) (round(x)<trunc(x)) and (abs(x)<=2) and (abs(y)<=2);

е)* (abs(x)<=1) > (abs(y)>=1);

ж) (sqr(x)+sqr(y)<=4) = (y<=x);

з) (abs(x)+abs(y)<=1) = (x*x+y*y>4);

и) succ(x>1) = (y<0)

2.15*. Записать на Паскале логическое выражение, зависящее от x и y, которое принимает значение *true*, только если точка с координатами (x,y) принадлежит заштрихованной области (см. рис. 1). (Пунктирная граница означает, что она не входит в область.)

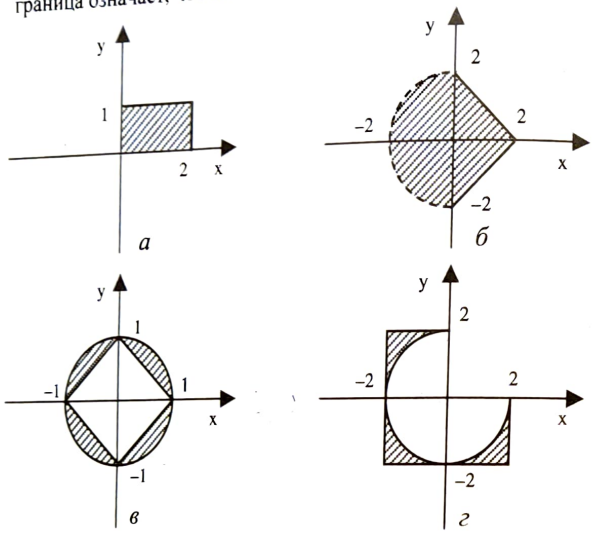


Рис. 1

2.16. Доказать тождества (a и b – логические величины):

а) not (a and b) ≡ (not a) or (not b);
б) not (a or b) ≡ (not a) and (not b);
в) a and (b or c) ≡ (a and b) or (a and c);
г) a or (b and c) ≡ (a or b) and (a or c);
д)* $a<=b$ ≡ not a or b;
е) a and b ≡ (a<true)<b;
ж) not a ≡ a<true

2.17. Преобразовать выражение к виду, не содержащему знаков отношения (a и b – логические величины):

а)* a<b; б) a=b; в) (a<b)=a

2.18. Если a=*true* и x=1, то какое значение получит логическая переменная d после выполнения оператора присваивания:

а) d:=x<2; б) d:=not a or odd(x); в) d:=ord(a)<>x

2.19. Выписать оператор присваивания, в результате выполнения которого логическая переменная t получает значение *true*, если выполняется указанное условие, и значение *false* иначе:

а) логическая переменная a имеет значение *true*; б) число x положительно;

в) целое число k нечетно;

г) p делится нацело на q (p и q – натуральные числа);

д) логические переменные a и b имеют одинаковые значения;

е) числа x, y, z равны между собой;

ж) из чисел x, y, z только два равны между собой;

з) уравнение $ax^2+bx+c=0$, где a, b и c могут равняться 0, имеет ровно один корень;

и) цифра 5 входит в десятичную запись трехзначного целого числа k

к) поля ($г1$, $в1$) и ($г2$, $в2$) шахматной доски имеют одинаковый цвет ($г1$, $в1$, $г2$ и $в2$ – целые от 1 до 8, обозначающие номера горизонтали и вертикали полей);

л) ферзь, расположенный на поле ($г1$, $в1$) шахматной доски, «бьет» поле ($г2$, $в2$).

3. ВВОД-ВЫВОД. ПРОСТЕЙШИЕ ПРОГРАММЫ

3.1. Ответить на следующие вопросы о процедурах ввода *read* и *readln*.

а) Почему в процедуре ввода *read*(v) параметр v обязательно должен быть переменной? Чем плохи, к примеру, обращения *read*(1) и *read*(2*x)?

б) Можно ли с помощью процедуры *read* ввести логическую величину?

в) Если в процедуре *read*(v) параметр v является целой переменной, то может ли пользователь программы задать для ввода не целое число, а вещественное? Что произойдет, если он неправильно записал число?

г) Если в процедуре *read*(v) параметр v является вещественной переменной, то может ли пользователь программы задать для ввода не вещественное число, а целое?

д) Верно ли, что вводимые числа должны записываться по правилам языка Паскаль? Можно ли, например, число записать как 1/2, а не как 0.5?

е) Перед вводимым числом можно указать пробелы. Что с ними происходит при вводе?

ж) Верно ли, что обращение *read*(v_1,v_2,…,v_n) эквивалентно последовательности обращений *read*(v_1); *read*(v_2); …; *read*(v_n)?

з) Между соседними вводимыми числами надо записать хотя бы один пробел. Зачем?

и) Что произойдет, если для процедуры *read*(*x*,*y*) пользователь программы вместо двух величин задал только одну?

к) Пусть для процедуры *read*(*x*) пользователь программы задал не одну величину, а две. Что произойдет при вводе с первой из них? А со второй?

л) Если в программе выполняются процедуры *read*(*x*,*y*); *readln*, а пользователь для ввода задал сразу четыре величины, то что произойдет с каждой из них?

м) Эквивалентно ли обращение *readln*(v_1,v_2,…,v_n) последовательности обращений *read*(v_1,v_2,…,v_n); *readln*? Описать действие процедуры *readln* в данном случае.

3.2*. Пусть *k* – целая, *x* – вещественная, а *b* – логическая переменные. Ниже слева указаны процедуры ввода, а справа – данные, которые пользователь задал для них. Определить, какие из этих процедур проработают без ошибок и какие значения они присвоят переменным.

а) read(k); read(x)	–25 3.14
б) read(k,x)	–25 3.14
в) read(x,k)	5 6
г) read(x,k)	5.0 6.0
д) read(k,k,k)	8 17 4
е) read(x)	e
ж) read(b)	true

3.3. Ответить на следующие вопросы о процедуре вывода *write*(*e*).

а) В процедуре *write*(*e*) параметр *e* может быть выражением. Каких типов? Что именно выводит процедура – само выражение или его значение?

б) Параметром *e* процедуры *write*(*e*) может быть непустая последовательность символов в кавычках. Что в таком случае выводит процедура?

в) С какой позиции экрана компьютера начинает выводить процедура *write* – с новой строки или с того места, где «остановилась» предыдущая процедура вывода?

г) Верно ли, что обращение *write*(e_1,e_2,…,e_n) эквивалентно последовательности обращений *write*(e_1); *write*(e_2); …; *write*(e_n)?

д) Сколько пробелов вставляет процедура *write* между соседними выведенными величинами для того, чтобы они не «слипались»?

3.4*. Пусть *k*=3 и *b*=*true*. Определить, что будет выдано на экран:

а) write(2*k+1);	б) write(not b);	в) write('k+1');
г) write('k=', k);	д) write(12, k);	е) write(12,' ', k);
ж) write(b, not b);	з) write(b, ' ', not b)	

3.5*. Процедура ввода *read* не выдает никакого приглашения к вводу, т.е. не выдает на экран компьютера текст, по которому пользователь программы мог бы узнать, что он должен начать ввод. Как реализовать такое приглашение (например, в виде символа >) при вводе значения для переменной *x*?

3.6. При обращении *write*(*e*:*n*) значение целочисленного выражения *n* задает ширину поля вывода, т.е. число очередных позиций экрана, в которых будет записано значение выражения *e*. Ответить на следующие вопросы о таком выводе.

а) Пусть значением *e* является целое число. Выводится ли знак «+»? Если ширина *n* больше длины записи числа, то к какому краю поля вывода «прижимается» число – к левому или правому? А если *n* меньше этой длины, то что выводится - всё число или только часть его?

б) Пусть *e* – последовательность из *k* символов, заключенная в кавычки. Если *n*>*k*, то к каком краю поля вывода «прижимаются» эти символы? А если *n*<*k*, то что выводится - все *k* символов или только *n* первых символов?

в) Верно ли, что вывод логической величины эквивалентен выводу *write*('*true*':*n*) или *write*('*false*':*n*)?

г) Если значение *e* - вещественное число, то оно выводится в экспоненциальной форме (с одной цифрой в целой части, с дробной частью и с десятичным порядком). Выводится ли в данном случае знак «+», или он заменяется на пробел, или он вообще никак не указывается? Если ширина *n* достаточно большая, то чем заполняются «лишние» позиции поля вывода – пробелами слева от числа или таким количеством нулей в дробной части числа, чтобы запись числа занимало всё поле вывода? А если ширина достаточно мала (например, *n*=1), то какие части числа обязательно выводятся? Происходит ли округление при отбрасывании «лишних» цифр в дробной части?

д) Обращение *write*(*e*) эквивалентно обращению *write*(*e*:*n*), в

котором берется стандартное значение *n*. Каково это значение для разных типов выражения *e*?

3.7*. Пусть *k*=123 и *n*=4. Указать текст (с пробелами), который будет выдан на экран:

а) write(k:5); б) write(k:1); в) write(k);
г) write(k+1:n+1); д) write(k, n); е) write(k:4, n);
ж) write(k, n:4); з) write(k:4, n:4)

3.8. Пусть *t=true*. Указать текст (с пробелами), который будет выдан на экран:

а) write('abc'); б) write('abc':5); в) write('abc':2);
г) write('a' , 'bc':5); д) write(t); е) write(t:5);
ж) write(not t : 5); з) write(t:2); и) write(t, t:5)

3.9*. Выписать обращение к процедуре *write* для вывода трех звездочек при условии, что между первой и второй из них должно быть 10 пустых позиций, а между второй и третьей – 17 пустых позиций.

3.10*. Пусть x=12.3456. Предполагая, что при выводе вещественного числа его порядок записывается 4 символами (буква E, знак и две цифры), указать текст (с пробелами), который будет выдан на экран:

а) write(x:10); б) write(–x:10); в) write(x:14);
г) write(–x:14); д) write(x:1); е) write(–x:1)

3.11. Обращение *write*(*e*:*n*:*m*) используется для вывода в очередных *n* позициях экрана вещественного числа *e* в форме с фиксированной точкой (без десятичного порядка) с *m* цифрами в дробной части. Ответить на следующие вопросы о таком выводе.

а) Может ли выражение *e* быть целого типа?

б) Выводится ли знак «+», или он заменяется на пробел, или вообще не указывается?

в) Все ли цифры целой части числа выводятся, если ширина *n* поля вывода меньше количества этих цифр?

г) Всегда ли выводится *m* цифр дробной части числа? А если *n*<*m*? А если в дробной части менее *m* цифр?

д) Происходит ли округление при отбрасывании «лишних» цифр в дробной части?

3.12*. Пусть *x*=12.3456. Указать текст (с пробелами), который будет выдан на экран:

а) write(x:10:3); б) write(–x:10:3); в) write(x:10:6);
г) write(–x:10:6); д) write(x:12:3); е) write(–x:12:3);

ж) write(x:1:3); з) write(–x:1:3); и) write(x:1:1)

3.13. Ответить на следующие вопросы о процедуре *writeln*.

а) Каково действие процедуры *writeln* (без параметров)?

б) Верно ли, что обращение *writeln*(x_1,x_2,…,x_n) эквивалентно последовательности обращений *write*(x_1,x_2,…,x_n); *writeln*? Когда при таком обращении процедура *writeln* делает перевод строки – до вывода первого параметра x_1 или после вывода последнего параметра x_n?

в)* Как будут расположены числа на экране в результате выполнения следующей последовательности процедур:

writeln; write(1, 2:2); writeln(3);
write(4:2, 5:2); writeln; writeln(6:2, 7:2) ?

3.14. Выписать последовательность обращений к процедуре *writeln* для вывода на экран следующей таблицы из целых чисел *a*, *b* и *c*:

a *b*
a *b* *c*
b *c*

при условии, что во всех строках правые цифры чисел *a*, *b* и *c* должны находиться соответственно в позициях 10, 20 и 30.

3.15. Ответить на следующие вопросы о Паскаль-программе.

а) Если в программе есть ввод, то обязательно ли в ее заголовке указывать имя *input* входного файла? А если есть вывод, то обязательно ли указывать имя *output* входного файла?

б) Являются ли имена *input* и *output* стандартными?

в) *integer*, *real* и *boolean* – это стандартные имена или служебные слова?

г) Можно ли одним и тем же именем обозначить разные объекты программы?

д) Любую ли переменную, используемую в программе, надо описывать в разделе переменных программы?

е) Эквивалентны ли разделы переменных

var x: real; y: real; и var x, y: real; ?

ж) Какую информацию извлекает транслятор из описания переменных и как он ее использует?

з) Какие значения получают переменные при описании?

и) Где в языке Паскаль ставятся точки с запятой – между соседними операторами или в конце каждого оператора? Например, надо ли ставить точку с запятой за последним оператором программы?

к) Верно ли, что выполнение программы (если не было ошибки) завершается только после выполнения ее последнего оператора?

л) Верно ли, что комментарии разрешено указывать только в тех местах программы, где может находиться пробел? Влияют ли как-нибудь комментарии на выполнение программы? Зачем они нужны?

3.16*. Какой результат будет выдан программой

```
program корни (input, output);
  var b, c, d: real;
begin
  read(b, c);
  d:=sqrt(sqr(b)–4*c);
  writeln('x1=', (–b+d)/2 : 1 : 2, '  x2=', (–b–d)/2 : 1 : 2)
end.
```

если в качестве исходных данных заданы числа 3.0 и 2.0?

3.17*. Написать программу, которая вводит два вещественных числа, вычисляет и выводит коэффициенты приведенного квадратного уравнения, корнями которого являются эти числа.

3.18. Какой результат будет выдан программой

```
program min (input, output);
  var x, y, m: integer;
begin
  read(x, y);
  m:=(x+y–abs(x–y)) div 2;
  writeln('min=', m)
end.
```

если в качестве исходных данных заданы числа 5 и –7?

3.19. Какой результат будет выдан программой

```
program less (input, output);
  var x: real;  t: boolean;
begin
  read(x);  t:=x<round(x);
  read(x);  t:=t and (x<trunc(x));
  writeln(t)
end.
```

если для ввода заданы числа 1.5 и –0.8?

3.20. Написать программу, которая выводит *true* или *false* в зависимости от того, имеют три заданных целых числа одинаковую четность или нет.

3.21. Ответить на следующие вопросы о константах.

а) В разделе констант можно давать имена различным данным

(числам и т.д.). С какой целью это делается?

б) Что именно можно указать в разделе констант *const C=…*; вместо многоточия: число без знака, число со знаком, непустая последовательность символов в кавычках, имя другой константы, знак и имя константы, любое выражение из констант?

в) В чем отличие констант от переменных?

г) В описании констант их типы не указываются. Как же определяются типы констант?

д) Почему не надо описывать константы *maxint*, *true*, *false*?

е) Верно ли, что любое (нестандартное) имя сначала должно быть описано в программе и лишь затем им можно пользоваться? Какое из имен *A* и *B* описывается в разделе констант *const A=B*; и какое используется?

3.22*. Найти и объяснить ошибки в каждом разделе констант:

а) const n=100; m=–1.7; k=n; n1=n+1;

б) const pi=3.14159; mpi=–pi; s=pi/2;

в) const asterisk=star; star='*'; prompt='input: ';

3.23*. Программа

```
program степени (output);
  const e=2.71828;
  var e2: real;
begin
  e2:=e*e;  writeln(e:10:5,e2:10:5, e*e2:10:5, e2*e2:10:5)
end.
```

выводит первые четыре степени числа *e*. Какие изменения (по возможности минимальные) надо внести в эту программу, чтобы она выводила первые четыре степени числа π?

3.24*. Написать программу, которая выдает результат *true* или *false* в зависимости от того, больше число e^{π} числа π^{e} или нет. (Числа *e* и π с точностью 10^{-5} описать как константы.)

3.25. Требуется вычислить периметр и площадь правильного 17-угольника, вписанного в окружность заданного радиуса.

Описать программу для решения этой задачи, причем описать так, чтобы минимальными изменениями данную программу можно было «настроить» на решение такой же задачи для 25-угольника.

3.26*. Найти и объяснить ошибки в каждой из следующих программ.

а)

```
program A (output);
  const d=5;
```

```
begin d:=sqr(d); writeln('d**2=',d) end.
```

б)
```
program B (input, output);
  const k=true;
  var x: real;
begin read(x); writeln(ord(x)=k) end.
```

в)
```
program V (input, output);
  var a, b, c: integer;
begin read(a, b); writeln((a+b+c)/3) end.
```

г)
```
program G (input, output);
  var x: real;
begin read(x); y:=sqrt(x)+1; writeln(y) end.
```

д)
```
program D (input, output);
  const B=2.5;
  var a, b, c: real;
begin read(a,c); writeln(a*c>b) end.
```

3.27. Найти и объяснить ошибки в следующей программе:

```
program ошибки (input, output);
  const π=3.14159;
  var a, b: integer;
begin read(A); d:=odd(π*0) and b>a; writeln(d) end.
```

3.28. Написать программу для решения следующей задачи.

а) Вычислить значение производной функции x^x в заданной точке a ($a>0$).

б) Для заданного числа a вычислить принадлежащий интервалу $(\pi, 2\pi)$ корень уравнения $ln(ctg\,x-1)=a$.

в) Вычислить дробную часть среднего геометрического трех заданных положительных чисел.

г) По заданным коэффициентам и правым частям уравнений системы

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$$

найти ее решение в предположении, что определитель системы не равен нулю.

д) Вычислить длину окружности, площадь круга и объем шара одного и того же заданного радиуса.

е) Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов.

ж) По заданным координатам трех вершин треугольника найти

его площадь и периметр.

з) По заданным длинам двух сторон треугольника и углу (в градусах) между ними найти длину третьей стороны и площадь этого треугольника.

и) Найти произведение цифр заданного четырехзначного числа.

к) Определить число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.

3.29. Для решения каждой из следующих задач написать программу, которая выводит *true* или *false* в зависимости от того, выполняется или нет указанное условие.

а) Для произвольных заданных вещественных чисел a, b и c определить, имеет ли уравнение $ax^2+bx+c=0$ хотя бы одно вещественное решение.

б) Для заданных чисел p, a и b ($a<b$) определить, имеет ли уравнение $arctg(2^x - |p|) = \sqrt{2}$ корень на отрезке $[a,b]$.

в) Определить, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

г) Определить, симметрична ли запись заданного четырехзначного числа.

д) Определить, есть ли цифра 0 среди первых трех цифр вдробной части заданного положительного вещественного числа.

е) Определить, есть ли среди цифр заданного трехзначного числа одинаковые.

ж) Даны три произвольных числа. Определить, можно ли построить треугольник с такими длинами сторон.

з) Даны координаты (как целые от 1 до 8) двух полей шахматной доски. Определить, может ли конь за один ход перейти с одного из этих полей на другое.

4. ОПЕРАТОРЫ: ПУСТОЙ, СОСТАВНОЙ, УСЛОВНЫЙ, ПЕРЕХОДА

4.1. Ответить на следующие вопросы о пустом, составном и условном операторах.

а) В каких случаях используется пустой оператор? Как определить, есть ли в данном месте программы пустой оператор или нет?

б) Выполнение составного оператора *begin* S_1; S_2; …; S_n *end* эквивалентно выполнению последовательности операторов S_1; S_2; …; S_n. В чем же тогда выгода от заключения этой последовательности в операторные скобки *begin* и *end*?

в) В составном операторе не ставится точка с запятой перед *end*. Будет ли ошибкой, если ее все же поставить? Например, допустим ли оператор *begin* x:=1; y:=2; *end*?

г) Сколько операторов можно записать в условном операторе между служебными словами *then* и *else* и после *else*?

д) Допустимо ли следующее сокращение записи условного оператора: *if B else S*?

е) Можно ли в условном операторе ставить точку с запятой перед *else*? Например, допустима ли запись *if* x>0 *then* y:=1; *else* z:=2?

ж) Эквивалентны ли оператор

if B_1 *then* S_1 *else if* B_2 *then* S_2

и последовательность операторов

if B_1 *then* S_1; *if* B_2 *then* S_2?

з) К какому из двух *then* относится единственный *else* в условном операторе

if B_1 *then if* B_2 *then* S_1 *else* S_2?

4.2*. Есть ли в следующих текстах пустые операторы? Где именно?

а) if x>0 then x:=2 else; y:=x+1;
б) if odd(k) then else k:=0;
в) begin x:=2; end;
г) begin a:=true; ; b:=b or a end;
д) begin ; end;
е) begin end

4.3*. Определить, какие значения будут иметь переменные x и y после выполнения операторов:

а) if x>0 then x:=1;
б) if x>0 then x:=1 else y:=2;
в) if x>0 then x:=1; y:=2;
г) if x>0 then begin x:=1; y:=2 end;
д) if x>0 then x:=1 else begin y:=2; x:=3 end

если вначале эти переменные имели следующие значения:

1) x=4, y=4; 2) x=–4, y=4

4.4. Определить, какое значение будет иметь переменная x после выполнения операторов:

а)*if x>0 then x:=1 else if y>0 then x:=2 else x:=3;

б) if x>0 then x:=1; if y>0 then x:=2 else x:=3;
в) if x>0 then x:=1 else if y>0 then x:=2;
г) if x>0 then x:=1; if y>0 then x:=2;
д)* if x>0 then if y>0 then x:=1 else x:=2 else x:=3;
е)* if x>0 then if y>0 then x:=1 else x:=2;
ж) if x>0 then begin if y>0 then x:=1 end else x:=2

если вначале переменные x и y имели следующие значения:

1) x=4, y=4; 2) x=–4, y=4; 3) x=4, y=–4; 4) x=–4, y=–4

4.5. Записать указанное действие в виде одного условного оператора.

а)* $y = \begin{cases} \cos^2 x & \text{при } 0 < x < 2 \\ 1 - \sin x^2 & \text{иначе} \end{cases}$

б)* Переменной x присвоить корень уравнения $arcsin(1+ln\ x)=a$, если такой существует.

в) Перераспределить значения переменных x и y так, чтобы в x оказалось большее из этих значений, а в y – меньшее.

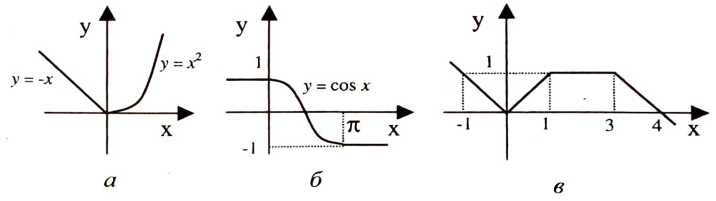
г)* $d=\max(a,b,c)$

д)* $z = \begin{cases} \max(x, y) & \text{при } x < 0 \\ \min(x, y) & \text{при } x \ge 0 \end{cases}$

е) Переменной k присвоить номер четверти плоскости, в которой находится точка с координатами x и y ($x \cdot y \neq 0$).

ж) Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой; присвоить номер этого числа переменной n.

4.6. Вычисление $y = f(x)$, где функция $f(x)$ задана графиком (см. рис. 2), описать в виде одного оператора.



а *б* *в*

Рис. 2

4.7. Записать последовательность операторов для решения указанной задачи.

а)* По номеру y ($y>0$) некоторого года определить c - номер его столетия (учесть, что, к примеру, началом XXI столетия был 2001, а не 2000 год).

б) $u = \dfrac{\max^2(x,y,z) - 2^x \cdot \min(x,y,z)}{\sin 2 + \max(x,y,z) / \min(x,y,z)}$

в)* Если уравнение $ax^2+bx+c=0$ ($a\neq 0$) имеет вещественные корни, то логической переменной t присвоить значение *true*, а переменным $x1$ и $x2$ - сами корни, иначе же переменной t присвоить *false*, а значения переменных $x1$ и $x2$ не менять.

г) Считая, что стандартные функции *sin* и *cos* применимы только к аргументам из отрезка $[0, \pi/2]$, вычислить $y=\sin x$ для произвольного числа x.

д)* Значения переменных a, b и c поменять местами так, чтобы оказалось $a\geq b\geq c$.

4.8. Найти и объяснить ошибки:

а)
```
if 1<x<2 then x:=x+1;  y:=0;
         else x:=0;  y:=y+1;
```

б)
```
if 1<x and x<2
      then begin x:=x+1;  y:=0 end;
      else begin x:=0;  y:=y+1 end
```

4.9*. Логической переменной b требуется присвоить значение *true*, если числа x и y равны, и значение *false* иначе. Определить, какой из следующих операторов правильно решает эту задачу:

а) if x=y then b:=true else b:=false;

б) b:=x=y

4.10*. Записать условный оператор, который эквивалентен оператору присваивания

x:=a or b and c

(все переменные - логические) и в котором не используются логические операции (например, оператору x:=*not* a эквивалентен оператор *if* a *then* x:=*false else* x:=*true*).

4.11. Записать оператор присваивания, эквивалентный условному оператору

if a then x:=b else x:=c

где все переменные - логического типа.

4.12. Написать программу для решения указанной задачи.

а) Для заданного числа a найти корень уравнения $f(x)=0$, где

$$f(x) = \begin{cases} 2ax + |\ a-1\ | & \text{при } a>0 \\ \dfrac{e^x}{\sqrt{1+a^2}} - 1 & \text{иначе} \end{cases}$$

б) Дано число x. Вывести в порядке возрастания числа $ch\ x$, $1+|x|$ и $(1+x^2)^x$.

в) Даны числа a_1, b_1, c_1, a_2, b_2, c_2. Вывести координаты точки пересечения прямых, описываемых уравнениями $a_1x+b_1y=c_1$ и $a_2x+b_2y=c_2$, либо сообщить, что эти прямые совпадают, не пересекаются или вовсе не существуют.

г) Даны числа a, b и r ($a\neq 0$, $r>0$). Если прямая $y=ax+b$ и окружность $x^2+y^2=r^2$ пересекаются, то вывести координаты всех точек их пересечения, а иначе сообщить, что они не пересекаются.

д) Даны числа a, b и c ($a\neq 0$). Найти вещественные корни уравнения $ax^4+bx^2+c=0$. Если корней нет, то сообщить об этом.

е) Даны произвольные числа a, b и c. Если нельзя построить треугольник с такими длинами сторон, то вывести 0, иначе вывести числа 3, 2 или 1 в зависимости от того, равносторонний это треугольник, равнобедренный или какой-либо иной.

ж) Даны целые числа h, m, s и t ($0 \leq h \leq 23$, $0 \leq m$, $s \leq 59$, $t>0$). Трактуя их как h часов, m минут и s секунд некоторого момента времени суток, определить час, минуту и секунду момента времени через t секунд (учесть смену суток).

з) Дано целое k от 1 до 180. Определить, какая цифра находится в k-й позиции последовательности

10111213...9899,

в которой выписаны подряд все двухзначные числа.

и) Дано натуральное k. Определить k-ю цифру в последовательности

11010100010000100000...,

в которой выписаны подряд степени 10.

4.13. Ответить на следующие вопросы о метках и операторе перехода.

а) Верно ли, что в качестве метки можно использовать любое неотрицательное целое число?

б) 005 и 5 – это разные метки?

в) Можно ли оператор метить сразу несколькими метками?

г) Допустим ли случай, когда меткой помечен какой-то опера-

тор, но в программе нет оператора перехода на эту метку?

д) Обязательно ли в разделе меток перечислять все метки, которыми помечены операторы программы?

е) В каком порядке должны перечисляться метки в разделе меток?

ж) Какие из следующих операторов относятся к сложным операторам: присваивания, пустой, составной, условный, перехода?

з) Можно ли по оператору перехода входить внутрь сложного оператора? А переходить из одной ветви (например, из части *then*) условного оператора в другую ветвь (в часть *else*)?

и) Можно ли выходить из сложного оператора по оператору перехода на метку, расположенную вне этого сложного оператора?

4.14*. Если n=3, то какое значение будет иметь переменная f после выполнения следующего составного оператора?

```
begin
  f:=1; i:=2;
1: if i>n then goto 9;
  f:=f*i; i:=i+1; goto 1;
9:
end
```

4.15*. Выписать последовательность операторов для решения каждой из следующих задач.

а) Вычислить c – наибольший общий делитель натуральных чисел a и b.

б) Найти u – первый отрицательный член последовательности $\cos(ctg\ n)$, n = 1, 2, 3, ...

в) Вычислить $p=(1-\frac{1}{2^2})(1-\frac{1}{3^2})\ \ \ldots\ \ (1-\frac{1}{n^2})$, $n>2$.

г) Вычислить $y=\cos(1+\cos(2+\ldots+\cos(39+\cos 40)\ldots))$.

4.16*. Программа. Дано 50 вещественных чисел. Найти величину наибольшего из них.

4.17. Программа. Дано целое n>0, за которым следует n вещественных чисел. Определить, сколько среди них отрицательных.

4.18. Программа. Подсчитать, сколько раз цифра 5 входит в десятичную запись заданного неотрицательного целого числа.

4.19. Найти и объяснить ошибки в следующих операторах:

а) begin 77: if |x|<1 then goto 5
  begin x:=x/2; goto 77; 5: y:=sin(x) end
end;

б) 34: if 0<x<2
  then if y<1 then goto 34 else goto 15
  else 15: y:=sqr(y)

4.20. Найти и объяснить ошибки в программе:

```
program errors (input, output);
 const L=18;
 label 18, 8;
 var x, y: integer;
begin
  y:=0;
8: read(x); if x<0 then goto L; y:=y+x; goto 18;
L: writeln(y)
end.
```

4.21*. Согласно правилам языка Паскаль, выполнение программы заканчивается только после выполнения ее последнего оператора. А что делать, если ее надо завершить в «середине»?

Описать программу, которая вводит 20 чисел и, если все они положительны, вычисляет их среднее геометрическое. Но если хотя бы одно из чисел меньше или равно 0, то остальные числа не вводить и сообщить об ошибке.

4.22. Программа. Ввести последовательность (возможно, пустую) из натуральных чисел, за которой следует 0, и вывести *true*, если эта последовательность является одной из следующих:

1; 1 1; 1 1 1; 1 5; 5

и вывести *false*, если это любая иная последовательность.

5. ОПЕРАТОР ЦИКЛА

5.1. Ответить на следующие вопросы об операторе цикла *while B do S*.

а) Что происходит сначала – вычисление логического выражения B или выполнение оператора S? Почему этот оператор цикла называют «с предусловием»?

б) Что описывает логическое выражение B – условие, при котором надо продолжать выполнение цикла, или условие выхода из цикла?

в) Что является телом следующего *while*-цикла (т.е. какие операторы будут в нем повторяться):

```
while x>y do x:=2*x; y:=y/2 ?
```

Что делать, если в *while*-цикле надо повторять не один оператор, а несколько?

г)* Может ли быть так, что тело *while*-цикла (оператор S) ни разу не будет выполнено? Определить значение переменной k после выполнения оператора

while k>=10 do k:=k div 10

если вначале эта переменная имела следующее значение:

1) 5; 2) 10; 3) 429

5.2. Ответить на следующие вопросы об операторе цикла *repeat* $S_1; S_2; \ldots; S_n$ *until* B.

а) Что происходит сначала – выполнение операторов S_i или вычисление логического выражения B? Почему этот оператор цикла называют «с постусловием»?

б) Что описывает логическое выражение B – условие, при котором надо продолжать выполнение цикла, или условие выхода из цикла?

в)* Может ли быть так, что тело *repeat*-цикла (операторы S_i) ни разу не будет выполнено? Определить значение переменной k после выполнения оператора

repeat k:=k div 10 until k<10

если вначале эта переменная имела следующее значение:

1) 5; 2) 10; 3) 429

г) Эквивалентны ли следующие операторы:

while B do S и repeat S until not (B) ?

5.3. Ответить на следующие вопросы об операторе цикла с параметром: *for* v:=H *to* K *do* S и *for* v:=K *downto* H *do* S.

а) Эквивалентно ли выполнение оператора цикла

for i:=1 to 4 do S

выполнению следующей последовательности операторов:

i:=1; S; i:=2; S; i:=3; S; i:=4; S

а выполнение оператора цикла

for i:=4 downto 1 do S

выполнению последовательности операторов:

i:=4; S; i:=3; S; i:=2; S; i:=1; S ?

б) Что является телом следующего *for*-цикла (т.е. какие операторы будут в нем повторяться):

for n:=1 to 4 do x:=2*x; y:=y/2 ?

Что делать, если в *for*-цикле надо повторять не один оператор, а несколько?

в) Какого типа может быть параметр *for*-цикла (переменная v): целого, вещественного, логического?

г) Обязательно ли выражения H и K должны быть того же типа, что и параметр цикла?

д)* Верно ли, что выражения H и K вычисляются только один раз (при входе в цикл) и затем уже не перевычисляются? Сколько раз будет выполняться тело следующего цикла:

m:=3; for i:=1 to m do m:=m+1 ?

е) Сколько раз будет выполняться тело *for*-цикла, если $H>K$? А если $H=K$? Например, сколько раз выполнится тело следующего цикла:

for i:=1 to m do f:=f*i

если m=0 и если m=1?

ж) Можно ли в теле *for*-цикла менять значение параметра цикла? Допустим ли, например, следующий оператор цикла:

for j:=1 to 5 do begin x:=2*x; if x>100 then j:=5 end ?

з)* Какое значение по завершению цикла

for v:=H to K do S

будет иметь параметр цикла v: значение K, значение $succ(K)$ или неопределенное значение? А если осуществлен досрочный выход (по оператору перехода) из *for*-цикла, то какое значение будет у параметра цикла?

Определить, какое значение получит переменная m после выполнения операторов

for j:=1 to 3 do begin x:=2*x; if x>=16 then goto 1 end;
1: m:=j

если вначале переменная x имела следующее значение:

1) 10; 2) 7; 3) 1; 4) 2

5.4*. Вычисление f =10! описать каждым из вариантов оператора цикла.

5.5*. Выписать фрагмент программы для решения указанной задачи и обосновать, почему был выбран тот или иной вариант оператора цикла.

а) Вычислить c – наибольший общий делитель натуральных чисел a и b.

б) Найти u – первый отрицательный член последовательности $cos(ctg\ n)$, n=1, 2, 3, ...

в) Вычислить $p = (1-\frac{1}{2^2})(1-\frac{1}{3^2}) \quad \ldots \quad (1-\frac{1}{n^2}), \quad n > 2.$

г) Вычислить $y=cos(1+cos(2+ \ldots +cos(39+cos\ 40) \ldots))$.

5.6. Выписать фрагмент программы для решения указанной задачи.

а) Для целого k ($k\geq 0$) найти x – k-й член последовательности

$\{x_n\}$, где $x_0=1$, $x_n=nx_{n-1}+1/n$ при $n\geq 1$.

б)* Вычислить $s=1-\frac{1}{2}+\frac{1}{3}- \ \dots \ +(-1)^{n-1}\frac{1}{n}, \quad n\geq 1.$

в)* Среди степеней числа 10 (1, 10, 100, …) найти p – первую (наименьшую), которая больше или равна числу x ($x>0$).

г) Найти y – первое из чисел $\sin x$, $\sin\sin x$, $\sin\sin\sin x$, ..., меньшее по модулю 10^{-4}.

д)* Подсчитать s – сумму цифр неотрицательного целого числа n.

е) Подсчитать k – количество цифр неотрицательного целого числа n.

ж) Найти d – старшую (левую) цифру неотрицательного целого числа n.

з) Определить, сколько раз (k) цифра 8 встречается среди первых пяти цифр дробной части положительного вещественного числа x.

и) Логической переменной t присвоить значение *true*, если натуральное число k есть степень 10 (1, 10, 100, …), и значение *false* иначе.

к) Получить целое число k добавлением слева цифры 6 к десятичной записи натурального числа n (например, k=6107 при n=107).

5.7. Описать программу для решения указанной задачи.

а) Дано n вещественных чисел (n=50). Найти максимум этих чисел.

б)* Дано n вещественных чисел (n=100). Вычислить разность между максимумом и минимумом этих чисел.

в)* Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить порядковый номер наименьшего из этих (ненулевых) чисел.

г) Дано n вещественных чисел (n=30). Определить порядковый номер того из них, который наиболее близок к своему номеру. (Считать, что такое число единственное.)

д) Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить, сколько раз в этой последовательности встречается наибольшее из этих чисел.

е) Найти наибольшую цифру в записи заданного неотрицательного целого числа.

5.8*. Используя тождество:

$a_0x^n+a_1x^{n-1}+a_2x^{n-2}+ \dots +a_{n-2}x^2+a_{n-1}x+a_0 \equiv$

$((\dots((a_0x+a_1)x+a_2)x+ \dots +a_{n-2})x+a_{n-1})x+a_0$,

вычислить по схеме Горнера:

а) $y=x^{10}+2x^9+3x^8+ \dots +10x+11$;

б) $y=11x^{10}+10x^9+9x^8+ \dots +2x+1$

5.9. Программа. Даны натуральное число n и вещественные числа t, a_0, a_1, …, a_n. Вычислить значение многочлена

$a_0x^n+a_1x^{n-1}+ \dots +a_{n-1}x+a_n$

и его производной в точке $t^2+0.5$

5.10. Найти k – число, полученное выписыванием в обратном порядке цифр неотрицательного целого числа n. (Например, если n=5074, то k=4705=$4\cdot10^3+7\cdot10^2+0\cdot10+5$.)

5.11. Целочисленный параметр *for*-цикла может меняться только с шагом ±1. А что делать, если по алгоритму этот параметр должен меняться с другим шагом?

Выписать фрагмент программы для решения указанной задачи.

а)* Вычислить $y=2\cdot4\cdot \dots \cdot(2n)$, $n>0$.

б) Вычислить $y=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+ \ \dots \ -\frac{1}{83}$

в) Вычислить $y=\frac{(1-x)(3-x)(5-x) \ \dots \ (39-x)}{(2-x)(4-x)(6-x) \ \dots \ (40-x)}$

г) Вычислить $y=\sqrt{3+\sqrt{6+\dots+\sqrt{96+\sqrt{99}}}}$

д) Вычислить $y=n!!=\begin{cases}1\cdot3\cdot5\cdot\dots\cdot n, & \text{если } n \text{ нечетно}\\ 2\cdot4\cdot6\cdot\dots\cdot n, & \text{если } n \text{ четно}\end{cases}$

е)* Вычислить $y=\sin 1+\sin 1.1+\sin 1.2+\dots+\sin 2$

5.12. Программа. Вывести на экран таблицу значений функций $\sin x$ и $\cos x$ на отрезке [0,1] с шагом 0.1 в следующем виде (на каждое число отводить 6 позиций строки):

```
   x    sin(x)   cos(x)
---------------------
0.0000 0.0000  1.0000
0.1000 0.0998  0.9950
   ...
1.0000 0.8415  0.5403
```

5.13. Программа. Приближенно вычислить интеграл

$\int_0^{\pi}\ln(2+\sin x)dx$,

используя формулу прямоугольников при n=100:

$$\int_a^b f(x)dx \approx h\cdot[f(x_1)+f(x_2)+\ldots+f(x_n)],$$

где $h=(b-a)/n$, $x_i=a+ih-h/2$.

5.14. Программа. Даны вещественные числа c и d ($c<d$). Приближенно вычислить интеграл

$$\int_c^d \cos e^x dx,,$$

используя формулу трапеций при n=80:

$$\int_a^b f(x)dx \approx h\cdot[f(a)/2+f(a+h)+ f(a+2h)+\ldots+f(b-h)+f(b)/2],$$

где $h=(b-a)/n$.

5.15*. При целочисленном параметре *for*-цикла начальное и конечное значение этого параметра должны быть целыми числами. А что делать, если по алгоритму они вещественные?

Выписать фрагмент программы для вычисления s – суммы квадратов всех целых чисел, попадающих в промежуток $(\ln x, e^x]$, $x>1$.

5.16. Подсчитать k – количество делителей натурального числа n из отрезка $[2, \sqrt{n}\,]$.

5.17. Вычислить k – количество точек с целочисленными координатами, попадающих в круг радиуса R ($R>0$) с центром в начале координат.

5.18. var k, i: integer; x, y: real;
Найти и объяснить ошибки в каждом из следующих фрагментов программы:

а) y:=0; for x:=0.1 to 0.9 do y:=y+sin(x);
б) k:=81; y:=1; for i:=1 to sqrt(k) do y:=2*y;
в) k:=0; for i:=1 to 9 do k:=k+sqr(i); k:=k*i;
г) k:=1; for i:=1 to 64 do begin i:=2*i; k:=k+i end

5.19. Вычислить:

а)* $y = \cos x+\cos x^2+\cos x^3+\ldots+\cos x^{30}$;
б)* $y = 1! + 2! + 3! + \ldots + n!$ ($n>1$);
в) $y=\sin x + \sin \sin x + \sin \sin \sin x + \ldots$ (последнее слагаемое – то, которое первым стало по модулю меньшее 10^{-4}).

5.20. Используя из стандартных функций только *abs*, вычислить с точностью $eps>0$:

а)* $y = e^x = 1+\frac{x}{1!}+\frac{x^2}{2!}+\ \ldots\ +\frac{x^n}{n!}+\ \ldots;$

б) $y = sh\,x = x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ \ldots\ +\frac{x^{2n+1}}{(2n+1)!}+\ \ldots;$

в) $y = \cos x = 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ \ldots\ +(-1)^n\frac{x^{2n}}{(2n)!}+\ \ldots;$

г) $y = \ln(1+x) = x-\frac{x^2}{2}+\frac{x^3}{3}-\ \ldots\ +(-1)^{n-1}\frac{x^n}{n}+\ \ldots\ (|x|<1);$

д) $y = arctg\,x = x-\frac{x^3}{3}+\frac{x^5}{5}-\ \ldots\ +(-1)^n\frac{x^{2n+1}}{2n+1}+\ \ldots\ (|x|<1)$

Считать, что требуемая точность достигнута, если очередное слагаемое по модулю меньше *eps*, – все последующие слагаемые можно уже не учитывать.

5.21. Числа Фибоначчи (f_n) определяются следующими формулами:
$f_0=f_1=1$; $f_n=f_{n-1}+f_{n-2}$ при n = 2, 3, ...

а)* Определить f – 40-е число Фибоначчи.
б) Найти f – первое число Фибоначчи, большее m ($m>1$).
в) Вычислить s – сумму всех чисел Фибоначчи, которые не превосходят 1000.

5.22*. С заданной точностью $eps>0$ вычислить x – наименьший положительный корень уравнения $x-tg\,x=0$, используя метод деления отрезка пополам.

5.23. Пусть функция $f(x)$ непрерывна на отрезке $[a,b]$ и $f(a)>0$, $f(b)<0$. Пусть уравнение $f(x)=0$ имеет только один корень на этом отрезке. Требуется найти данный корень x с точностью $eps>0$.

Нередко дается следующее решение этой задачи методом деления отрезка пополам:

```
repeat
  c:=(a+b)/2;
  if f(c)>0 then a:=c else b:=c
until abs(f(c))<eps;
x:=c
```

Правильное ли это решение?

(*Подсказка*: рассмотреть функцию $f(x)=eps\cdot(0.5-x)$ на отрезке $[-1,1]$ при eps=0.01.)

5.24. Программа. Дано *eps*>0. С точностью *eps* найти единственный корень уравнения

$$\pi x^3 - ex^2 + (2e+1)x + \pi^2 = 0$$

5.25. Каждый из следующих фрагментов программы преобразовать к виду, не содержащему оператор перехода.

а)*
```
for k:=1 to 10 do
    begin if x>y then goto 5; x:=2*x; y:=y/2 end;
k:=11;
5:
```

б)*
```
for k:=1 to 10 do
    begin x:=2*x; y:=y/2; if x>y then goto 5 end;
k:=11;
5:
```

в)
```
for k:=1 to 10 do
    begin x:=2*x; if x>y then goto 5; y:=y/2 end;
k:=11;
5:
```

5.26*. Показать, что составное целое число n ($n \ge 2$) обязательно имеет хотя бы один делитель из отрезка $[2, \sqrt{n}\,]$.

С учетом этого факта выписать фрагмент программы, в котором логической переменной *p* присваивается значение *true*, если целое n ($n \ge 2$) является простым числом, и значение *false* иначе, при следующем условии:

а) использовать оператор цикла с параметром и оператор перехода;

б) не использовать оператор перехода.

5.27. Если среди чисел $\sin x^n$ (n=1, 2, ..., 30) есть хотя бы одно отрицательное число, то логической переменной *t* присвоить значение *true*, а иначе – значение *false*.

а) Использовать оператор цикла с параметром и оператор перехода.

б) Не использовать оператор перехода.

5.28. Программа. Дана последовательность из n вещественных чисел (n=70). Определить, со скольких отрицательных чисел она начинается.

5.29. Программа. Даны целые числа a, b_1, b_2, …, b_n (n=30). Определить порядковый номер первого из чисел b_i, равного числу a. Если такого b_i нет, выдать ответ 0.

5.30. Программа. Дана последовательность из n целых чисел (n=25). Определить, является ли эта последовательность:

а)* возрастающей;

б) неубывающей;

в) арифметической прогрессией;

г) геометрической прогрессией.

В качестве ответа выдать слово *Да* или *Нет*.

5.31*. Требуется ввести 80 чисел $x_1, x_2, \ldots, x_{80}$ и вычислить

$$y=(x_1+x_2+\ldots+x_{10})\,(x_{11}+x_{12}+\ldots+x_{20})\ldots(x_{71}+x_{72}+\ldots+x_{80})$$

Какое из следующих решений этой задачи неправильное и почему?

а)
```
y:=1; s:=0;
for i:=1 to 8 do
 begin
  for j:=1 to 10 do begin read(x); s:=s+x end; y:=y*s
 end;
```

б)
```
y:=1;
for i:=1 to 8 do
 begin
  s:=0; for j:=1 to 10 do begin read(x); s:=s+x end;
  y:=y*s
 end;
```

5.32. Вычислить:

$$s = \frac{1}{1}\cdot\frac{1}{2}\cdot\ldots\cdot\frac{1}{20} + \frac{1}{21}\cdot\frac{1}{22}\cdot\ldots\cdot\frac{1}{40} + \ldots + \frac{1}{101}\cdot\frac{1}{102}\cdot\ldots\cdot\frac{1}{120}$$

5.33. Найти *min* – наименьшее из чисел $exp(sin(i+j^2))$, где i и j – целые, $0 \le i \le 40$, $0 \le j \le 25$.

5.34*. Найти *max* – наибольшее из чисел $cos(i^2+j^2)$, где i и j – целые, $1 \le i \le j \le 30$.

5.35. Программа. Даны вещественные числа $x_1, x_2, \ldots, x_{55}$. Вычислить величину

$$x_1(x_2+x_3)(x_4+x_5+x_6)(x_7+x_8+x_9+x_{10})\ldots(x_{46}+x_{47}+\ldots+x_{55})$$

5.36*. Определить k – количество трехзначных натуральных чисел, сумма цифр которых равна n ($1 \le n \le 27$). Операции деления (*/*, *div* и *mod*) не использовать.

5.37. Вывести на экран в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр. Операции деления (*/*, *div* и *mod*) не использовать.

5.38. Логической переменной *t* присвоить значение *true* или *false* в зависимости от того, можно или нет представить натуральное число

n в виде суммы трех квадратов: $n=a^2+b^2+c^2$, где a, b и c – натуральные числа, $a \geq b \geq c$.

5.39. Программа. Для логической функции

$F = (A \text{ and } B) \text{ or not } (B \text{ or } C)$

вывести на экран таблицу истинности в следующем виде:

```
-----------------------
  A     B     C     F
-----------------------
 true  true  true  true
 true  true false  true
 true false  true false
   . . .
false false false  true
-----------------------
```

5.40. Программа. Дано n вещественных чисел (n=20). Определить, сколько из них больше своих «соседей», т.е. предыдущего и последующего чисел.

5.41. Программа. Дана непустая последовательность ненулевых целых чисел, за которой следует 0. Определить, сколько раз в этой последовательности меняется знак. (Например, в последовательности 1, –34, 8, 14, –5 знак меняется 3 раза.)

5.42. Программа. Даны натуральное число n и вещественные числа $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$. Рассматривая пары x_i, y_i как координаты точек на плоскости, определить радиус наименьшего круга (с центром в начале координат), внутрь которого попадают все эти точки.

5.43. Программа. Даны целое n>2 и вещественные числа $a_1, b_1, \ldots, a_n, b_n$ ($a_i<b_i$). Рассматривая пары a_i и b_i как левые и правые концы отрезков на одной и той же прямой, определить концы отрезка, являющегося пересечением всех этих отрезков. Если такого отрезка нет, то сообщить об этом.

5.44. Программа. Дано n вещественных чисел (n=80). Найти порядковый номер того из них, которое ближе всего к какому-нибудь целому числу.

5.45. Программа. Дано не менее трех различных натуральных чисел, за которыми следует 0. Определить три наибольших числа среди них.

5.46. Дана неубывающая последовательность из n целых чисел (n=30). Определить количество различных элементов в этой последовательности.

5.47. Дана последовательность из n целых чисел (n=50). Выявить отрезки возрастания в этой последовательности и вывести каждый из них на экран с новой строки.

5.48. Программа. Дана последовательность из n целых чисел (n=100). Определить количество элементов в самой длинной подпоследовательности из подряд идущих нулей.

5.49. Программа. Даны целое n>1 и вещественные числа $x_1, x_2, \ldots, x_n$. Вычислить:

$$M = \frac{\sum x_i}{n}, \quad D = \sqrt{\frac{\sum (x_i - M)^2}{n-1}}$$

5.50. Программа. Дана непустая последовательность положительных вещественных чисел $x_1, x_2, \ldots, x_n$ (n заранее не известно), за которыми следует отрицательное число. Вычислить величину

$$nx_1+(n-1)x_2+ \ldots +2x_{n-1}+x_n.$$

5.51. Программа. Определить, является ли заданное натуральное число палиндромом, т.е. таким, запись которого читается одинаково слева направо и справа налево. (*Замечание*: числа типа 010 или 0220 не считать палиндромами.)

5.52. Программа. Найти все целые корни уравнения $ax^3+bx^2+cx+d=0$, где a, b, c и d – заданные целые числа, причем $a \neq 0$ и $d \neq 0$. (*Замечание:* целыми корнями могут быть только положительные и отрицательные делители коэффициента d.)

5.53. Программа. Дано 10 натуральных чисел. Найти их наибольший общий делитель.

5.54. Программа. Дано целое n>2. Вывести на экран все простые числа из диапазона $[2, n]$.

5.55. Программа. Определить, является ли заданное натуральное число совершенным, т. е. равным сумме всех своих (положительных) делителей, кроме самого этого числа (например, число 6 совершенно: 6=1+2+3).

5.56. Программа. Найти все простые делители заданного натурального числа.

6. СИМВОЛЬНЫЙ ТИП

6.1. Ответить на следующие вопросы о символьном типе *char*.

а) *char* – это стандартное имя или служебное слово?

б) Зачем в тексте программы символы нужно указывать в кавычках. Есть ли разница между *a* и *'a'*, между 5 и *'5'*, между + и *'+'*?

в) Каким образом можно указать кавычку как символ?

г) Сколько всего символов входит в тип *char* – 256 или другое количество?

д) Обязательно ли в тип *char* входят цифры, латинские буквы и русские буквы?

е) Равны ли как символы одноименные большие и малые латинские буквы? Например, верно ли, что *'w'='W'*?

ж) В типе *char* символы упорядочены. Как именно они упорядочены? Какой символ меньше всех других? А больше всех других? Что верно: *'9' < 'a'* или *'9' > 'a'*?

з) Упорядочены ли в типе *char* цифры по возрастанию их числовых значений? Может ли между символами *'0'* и *'1'* находиться другой символ? Верно ли, что если *c* – символ и *'0'≤c≤'9'*, то *c* – цифра?

и) Упорядочены ли в типе *char* малые (большие) латинские буквы по алфавиту? Может ли между символами *'g'* и *'h'* (между *'G'* и *'H'*) находиться другой символ? Верно ли, что если *c* – символ и *'a'≤c≤'z'*, то *c* – малая латинская буква?

к) Если русские буквы входят в тип *char*, то обязательно ли они упорядочены по алфавиту?

л) Каждому символу поставлен в соответствие порядковый номер (код) в типе *char*. С какого числа начинается эта нумерация – с 0 или с 1?

6.2. Ответить на следующие вопросы о функции *ord*(*c*) при символьном параметре.

а) Верно ли, что значением *ord*(*c*) является порядковый номер символа *c* в типе *char*? Каково минимальное значение функции *ord*? А максимальное?

б) Всегда ли выполняется равенство *ord('0')=0*?

в) Верно ли, что *ord(c)<ord(d)* тогда и только тогда, когда *c<d*?

г) Если *k* – цифра (от 0 до 9), то верно ли равенство *k=ord('k')–ord('0')*?

д) Всегда ли верно равенство *ord('h')=ord('g')+1*?

6.3. Ответить на следующие вопросы о функции *chr*(*k*).

а) Что обозначают в языке Паскаль имена *char* и *chr*?

б) Верно ли, что значением *chr*(*k*) является символ с порядковым номером *k* в типе *char*?

в) При любом ли целом *k* определена функция *chr*? Например, чему равно *chr*(–1)?

г) Для любого ли символа *c* выполняется равенство *chr(ord(c))=c*?

д) Для любого ли целого *k* выполняется равенство *ord(chr(k))=k*?

е) Если *k* – цифра (от 0 до 9), то верно ли равенство *'k'=chr(ord('0')+ k)*?

6.4. Ответить на следующие вопросы о функциях *succ* и *pred* при символьном параметре.

а) Верно ли, что значением *succ*(*c*) является символ, следующий по порядку за символом *c* в типе *char*? Для любых ли символов определена эта функция?

б) Верно ли, что значением *pred*(*c*) является символ, предшествующий по порядку символу *c* в типе *char*? Для любых ли символов определена эта функция?

в) Всегда ли верно равенство *succ('4')= pred('6')*?

г) Для любого ли символа *c* верно равенство *succ(pred(c))=c*?

д) Для любого ли символа *c* верно равенство *pred(c)=chr(ord(c)–1)*?

6.5. Ответить на следующие вопросы об операциях над символами.

а) Какие выражения называются символьными?

б) Можно ли к символам применять арифметические операции? Например, допустима ли операция *'5'+1*?

в) Можно ли в отношениях сравнивать символьные выражения?

г) Может ли символьная переменная быть параметром *for*-цикла?

д) Можно ли вводить символы с помощью процедуры *read*?

е) Можно ли выводить значения символьных выражений с помощью процедуры *write*?

6.6*. Вычислить выражения:

а) succ(succ('0'))=pred('3');
б) ord('5')–ord('0');
в) '5'–'0';
г) ord('d')+2<ord('g');
д) 'r'>'q';
е) ord('w')<10;
ж) chr(25)>=chr(16);
з) ord(pred(chr(24)));

и) chr(pred(ord('5'))+5); к) pred(chr(0))

6.7*. Имеется символьная переменная *d*. Присвоить логической переменной *t* значение *true*, если выполняется указанное условие, и значение *false* иначе:

а) значением *d* является символ '*';

б) значением *d* является символ '*a*' или '*q*';

в) значением *d* является цифра.

6.8*. Определить значение символьной переменной *d* после выполнения операторов:

а) c:='+'; d:=c; б) c:='+'; d:='c'

6.9*. Вычислить *s* – сумму порядковых номеров всех символов, входящих в слово '*SUM*'.

6.10*. Вывести текст, образованный символами с порядковыми номерами 65, 71 и 69.

6.11*. Символьной переменной *next* присвоить цифру, следующую за цифрой, являющейся значением символьной переменной *dig*, считая при этом, что за '9' следует '0'.

6.12*. Логической переменной *b* присвоить значение *true*, если в типе *char* между символами '*a*' и '*z*' нет иных символов, кроме малых латинских букв, и значение *false* иначе.

6.13*. Вывести в отдельной строке экрана все символы между '*A*' и '*Z*', включая и эти буквы.

6.14. Используя только символьный вывод, вывести на экран таблицу следующего вида:

а)*	б)	в)
100...00	999...99	0123456789
020...00	088...88	1234567890
...	...	...
000...09	000...01	9012345678

Далее в этом параграфе под «текстом» понимается заданная во входном файле input последовательность (возможно, пустая) символов, за которой следует точка (в сам текст точка не входит).

6.15*. Программа. Вывести значение *true*, если в заданном тексте буква '*a*' встречается чаще, чем буква '*b*', и значение *false* иначе.

6.16. Программа. Если в заданный текст входит каждая из букв слова '*key*', тогда в качестве ответа вывести слово '*yes*', а иначе – слово '*no*'.

6.17. Программа. Проверить, правильно ли в заданном тексте расставлены круглые скобки (т.е. находится ли справа от каждой открывающей скобки соответствующая ей закрывающая скобка, а слева от каждой закрывающей – соответствующая ей открывающая). Ответ – *true* или *false*.

6.18*. Программа. Определить, является ли заданный текст правильной записью целого числа (возможно, со знаком).

6.19. Известно, что в заданный текст входит буква '*a*', причем не на последнем месте. Требуется вывести символ этого текста, который непосредственно следует за первым вхождением '*a*'.

Можно ли решить эту задачу следующим образом (*c* – символьная переменная):

repeat read(c) until c='a'; writeln(succ(c)) ?

Ответ обосновать.

6.20*. Программа. Подсчитать, сколько раз пара '*th*' входит в заданный текст.

6.21. Программа. Подсчитать число вхождений тройки '*abc*' в заданный текст.

6.22. Программа. Подсчитать, сколько раз комментарий входит в заданный текст, понимая под комментарием отрезок текста, который:

а) начинается с символа '{' и заканчивается первым же символом '}' или концом текста;

б) начинается с пары '(*' и заканчивается первой же парой '*)' или концом текста (отрезок '(*)' рассматривать как комментарий).

6.23. Программа. Вывести заданный непустой текст:

а)* удалив из него все цифры и удвоив знаки '+' и '–';

б)* удалив из него все знаки '+', непосредственно за которыми идет цифра;

в) удалив из него все буквы '*b*', непосредственно перед которыми находится буква '*c*';

г) заменив в нем все пары '*ph*' на букву '*f*'.

6.24. Программа. Вывести заданный текст, удалив из него лишние пробелы, т.е. из нескольких подряд идущих пробелов оставить только один.

6.25. Программа. Заданный текст вывести по строкам, понимая под строкой либо очередные 60 символов, если среди них нет запятой, либо часть текста до запятой включительно.

6.26. Программа. Дана непустая последовательность непустых слов из латинских букв; соседние слова отделены друг от друга запятой, за

последним словом – точка. Определить количество слов, которые:

а)* начинаются с буквы *'a'*;

б) оканчиваются буквой *'w'*;

в) начинаются и оканчиваются одной и той же буквой;

г) содержат хотя бы одну букву *'d'*;

д) содержат ровно три буквы *'e'*.

6.27*. Пусть значением *c* является цифра как символ (от '0' до '9'). Как получить *d* – эту же цифру, но как число (от 0 до 9)?

Выписать фрагмент программы, в котором целой переменной *k* присваивается число, составленное (слева направо) из цифр, являющихся значениями символьных переменных *c2*, *c1* и *c0*. (Например, *k*=805, если *c2*='8', *c1*='0' и *c0*='5').

6.28*. Пусть значением *d* является цифра как число (от 0 до 9). Как получить *c* – эту же цифру, но как символ (от '0' до '9')?

Выписать фрагмент программы, в котором символьным переменным *c2*, *c1* и *c0* присваиваются соответственно левая, средняя и правая цифры трехзначного числа *k*.

6.29*. Используя только символьный ввод, т.е. процедуру *read*(*c*), где *c* – символьная переменная, ввести непустую последовательность цифр, перед которой может находиться знак '+' или '–' и за которой следует пробел, и, получив соответствующее целое число, присвоить его целой переменной *k*.

6.30. Используя только символьный вывод, т. е. процедуру *write*(*c*), где *c* – символьный параметр, вывести значение целой переменной *k* (знак '+' не выводить).

6.31. Программа. Дано натуральное число *n*. Вывести в троичной системе счисления все целые числа от 0 до *n*.

6.32. Программа. Дана непустая последовательность неотрицательных целых чисел, записанных в семеричной системе счисления; между соседними числами – пробел, за последним – точка. Определить наибольшее из них и вывести его в десятичной системе счисления.

6.33. Программа. Дано неотрицательное целое число, записанное в восьмеричной системе счисления (за числом – пробел). Вывести это число в пятеричной системе счисления.

6.34. Программа. Дана последовательность символов, имеющая следующий вид: $d_1 \pm d_2 \pm \ldots \pm d_n$ (каждое d_i – цифра, n>1), за которой следует точка. Вычислить значение этой алгебраической суммы.

6.35. Решить предыдущую задачу при условии, что каждое d_i –

это целое без знака (непустая последовательность цифр).

6.36. Используя только символьный ввод, ввести заданное вещественное число (за ним – пробел), записанное по правилам языка Паскаль, и присвоить его вещественной переменной *x*.

6.37. Используя только символьный вывод, вывести вещественное число *x* в следующей форме:

$$\pm 0.d_1d_2\ldots d_9\mathrm{E}\pm p_1p_2\,,$$

где d_i, p_j – цифры, причем $d_1 \neq 0$ при $x \neq 0$.

6.38*. Имеются целые числа *k* и *n* от 1 до 79. Вывести в отдельной строке экрана:

а) символ '*' в *k*-й позиции;

б) символ '*' в *k*-й позиции и символ 'I' в *n*-й позиции (при *k*=*n* вывести только звездочку).

6.39. Программа. Вывести на экран график функции $y=x^2-1$ на отрезке [–1, 2] с шагом 0.25. Ось OX направить по вертикали вниз, а ось OY – по горизонтали вправо. В каждой строке экрана выводить «кусочек» оси OX (например, символ '|') и звездочку – в позиции, соответствующей очередному значению функции; ось OY не выводить. Примерный вид графика:

6.40. Программа. Даны натуральные числа *a* и *b* (*a*<*b*). Вывести на экран в точках *a*, *a*+1, …, *b* график функции Эйлера φ(*n*), вычисляющей количество целых чисел от 1 до *n*–1, взаимно простых с числом *n*.

6.41. Программа. В заданный текст произвольной длины (>0) входят только цифры и большие латинские буквы (за текстом – точка). Определить, удовлетворяет ли он следующему свойству:

а) текст является десятичной записью числа, кратного 9;

б) текст является записью четного числа в семеричной системе;

в) текст является десятичной записью числа, кратного 6;

г) текст является десятичной записью числа, кратного 4;

д) текст является шестнадцатеричной записью числа, кратного 5;

е) текст начинается с некоторой ненулевой цифры, за которой

следуют только буквы, и их количество равно числовому значению этой цифры;

ж) текст начинается с k букв ($1 \le k \le 9$), за которыми следует только один символ – цифра с числовым значением k;

з) текст совпадает с начальным отрезком ряда 0123456789 (например: 0, 01, 012);

и) текст совпадает с конечным отрезком ряда 0123456789 (например: 9, 89, 789);

к) текст совпадает с каким-то отрезком ряда 0123456789 (например: 2, 678);

л) текст состоит только из цифр, причем их числовые значения образуют арифметическую прогрессию (например: 2468, 741, 3);

м) текст содержит (помимо букв) только одну цифру, причем ее числовое значение равно длине текста;

н) сумма числовых значений цифр, входящих в текст, равна длине текста.

7. ПЕРЕЧИСЛИМЫЕ И ОГРАНИЧЕННЫЕ ТИПЫ. ОПЕРАТОР ВАРИАНТА

7.1. Ответить на следующие вопросы о разделе типов Паскаль-программы.

а) Для чего нужен раздел типов: чтобы определить новый тип данных, чтобы дать имя типу данных или для того и другого? Надо ли в разделе типов описывать стандартные типы данных (*integer*, *char* и т.д.)?

б) Пусть T и $T1$ – имена и пусть имеется раздел типов

```
type T=T1;
```

Определяется ли в таком разделе новый тип данных? Что здесь должно обозначать имя $T1$? Должно ли это имя быть уже описанным? Что будет обозначать имя T?

Если, к примеру, имеется раздел типов:

```
type int=integer;
```

то эквивалентны ли следующие описания переменных x и y?

1) var x, y: integer; 2) var x, y: int; 3) var x: int; y: integer;

в) Пусть T – имя, а KT – конструктор (определение, задание) типа и пусть имеется раздел типов

```
type T=KT;
```


Определяется ли в таком разделе новый тип данных? Каково назначение здесь конструктора типа KT? Что будет обозначать имя T?

г) Верно ли, что в языке Паскаль каждый конструктор типа определяет тип данных, который отличен от всех других типов?

д) Можно ли в Паскаль-программе определить новый тип данных, но не дать ему имени, т.е. допускается ли использование безымянных типов?

е) Пусть KT – конструктор некоторого типа данных. Какие из следующих фрагментов:

1)
```
type T=KT;
var x, y: T;
```

2)
```
type T=KT;
var x: T; y: T;
```

3)
```
var x, y: KT;
```

4)
```
var x: KT; y: KT
```

5)
```
type T=KT;
var x: T; y: KT;
```

описывают x и y как переменные одного типа, а какие нет и почему?

7.2. Ответить на следующие вопросы о перечислимом типе, определяемом конструктором типа $(\alpha, \beta, \ldots, \gamma)$.

а) Верно ли, что в качестве $\alpha, \beta, \ldots, \gamma$ можно указывать только имена и нельзя указывать числа, символы и служебные слова?

б) Верно ли, что в перечислимый тип входят имена $\alpha, \beta, \ldots, \gamma$ и только они?

в) Кто определяет, какие имена должны входить в перечислимый тип? Можно ли в один перечислимый тип включить и названия месяцев, и названия шахматных фигур?

г) Появление имен $\alpha, \beta, \ldots, \gamma$ в конструкторе перечислимого типа означает описание их как констант. Используя это правило, объясните, почему константы перечислимого типа не могут совпадать с именами других объектов программы (например, переменных) и не могут входить в другие перечислимые типы.

д) Порядок расположения имен в конструкторе перечислимого типа устанавливает отношение «меньше-больше» для этих имен. Как именно определяется это отношение? Например, что меньше: первое из указанных имен (α) или следующее за ним имя (β)?

е) Расположение имен в конструкторе перечислимого типа также определяет порядковые номера имен в этом типе. С какого числа начинается нумерация – с 0 или с 1?

ж) Если n – константа перечислимого типа, то каковы значения функций $succ(n)$, $pred(n)$ и $ord(n)$? Всегда ли в данном случае определены эти функции?

з) Константы перечислимых типов могут быть операндами от-

ношения. Должны ли они быть одного и того же типа или могут быть разных перечислимых типов?

и) Верно ли, что переменная перечислимого типа может быть параметром *for*-цикла?

к) Можно ли значения перечислимого типа вводить и выводить с помощью процедур *read* и *write*?

л) В каких из следующих фрагментов:

1) type T=(a,b);
   var x, y: T;

2) type T=(a,b);
   var x: T; y: (a,b);

3) var x, y: (a,b);

4) var x: (a,b); y: (a,b);

x и *y* будут описаны как переменные одного и того же типа, а в каких нет и почему?

Почему в программе нельзя дважды указывать один и тот же конструктор перечислимого типа?

7.3*. Имеются описания:

```
type сезон = (зима, весна, лето, осень);
var x, y: сезон;  t: (тепло, холодно);
```

Ответить на следующие вопросы.

а) Какие значения могут принимать переменные *x*, *y* и *t*? Допустимы ли присваивания:

1) x:=весна; 2) y:=x; 3) t:=тепло; 4) y:=t; 5) t:=жарко ?

б) Вычислить выражения:

1) весна<лето; 2) зима<=лето; 3) осень<зима;
4) весна<>тепло; 5) succ(весна); 6) pred(весна);
7) succ(осень); 8) pred(холодно); 9) ord(зима);
10) ord(осень); 11) ord(pred(тепло)); 12) pred(ord(тепло))

в) Допустим ли следующий заголовок *for*-цикла:

```
for x:=весна to осень do ?
```

г) Допустимы ли следующие операции ввода-вывода:

1) read(x); 2) write(лето); 3) write('лето'); 4) writeln('зимой ', t) ?

7.4*. Найти и объяснить ошибки в следующем разделе типов:

```
type буква = ('a', 'b', 'c', 'd');
     гласная = (a, e, i, o, u);
     согласная = (b..d, f, g);
     карта = (6, 7, 8, 9, 10, валет, дама, король, туз);
     корень = (1.00, 1.41, 1.73, 2.00);
     фигура = (слон, конь, ладья, ферзь);
     конь = (пегий, сивый, каурый);
```



```
     животное = (лев, слон, жираф);
     деление = (div, mod);
     лог = boolean;
```

7.5*.

```
type месяц = (янв, фев, мар, апр, май, июн, июл, авг,
              сен, окт, ноя, дек);
var d1, d2: integer;  m1, m2: месяц;  t: boolean;
```

Пусть пара *d1*, *m1* правильно задает число и месяц некоторого дня в году, а пара *d1*, *m2* – другого дня. Присвоить переменной *t* значение *true*, если дата *d1*, *m1* предшествует (в рамках года) дате *d2*, *m2*, и значение *false* иначе.

7.6.

```
type название = (шесть, семь, восемь, девять, десять,
                 валет, дама, король, туз);
     масть = (пики, трефы, бубны, червы);
var n1, n2: название;  m1, m2, k: масть;  b: boolean;
```

Переменной *b* присвоить значение *true* или *false* в зависимости от того, «бьет» или нет игральная карта с названием *n1* и масти *m1* карту с названием *n2* и масти *m2* при условии, что масть *k* является козырной.

7.7.

```
var m, m1: (янв, фев, мар, апр, май, июн, июл, авг,
            сен, окт, ноя, дек);
    k: integer;  {k>0}
```

Присвоить переменной *m1*:

а)* название месяца, следующего за месяцем *m* (с учетом, что за декабрем следует январь);

б) название месяца перед месяцем *m* (с учетом, что январю предшествует декабрь);

в)* название *k*-го месяца после месяца *m*;

г) название *k*-го месяца перед месяцем *m*;

7.8*. В языке Паскаль есть функция *ord*, позволяющая по константе перечислимого типа узнать ее порядковый номер в этом типе, но нет обратной функции. А что делать, если по номеру константы в перечислимом типе требуется узнать саму константу?

Выписать фрагмент программы для решения следующей задачи.

```
type масть = (пики, трефы, бубны, червы);
var n: integer;  m: масть;
```

По номеру масти *n* ($0 \le n \le 3$) определить *m* – название этой масти.

7.9*.

```
type страна = (Австрия, Болгария, Греция, Италия,
               Норвегия, Франция, Германия);
```

```
столица = (Вена, София, Афины, Рим,
           Осло, Париж, Берлин);
var st: страна;  cap: столица;
```

По значению переменной *st* (названию страны) присвоить переменной *cap* название столицы этой страны.

7.10.
```
var P: (ада, алгол, бейсик, джава, модула2,
        лисп, паскаль, пролог, си, фортран);
    A: (ada, algol, basic, java, modula2, lisp,
        pascal, prolog, C, fortran);
```

По *P* – русскому названию языка программирования присвоить переменной *A* английское название этого языка.

7.11.
```
type название = (нуль, один, два, три, четыре, пять);
var  d: '0'..'5';  n: название;
```

По символу-цифре *d* присвоить переменной *n* название этой цифры.

7.12.
```
type нота = (до, ре, ми, фа, соль, ля, си);
     интервал = (секунда, терция, кварта, квинта,
                 секста, септима);
var n1, n2: нота;  i: интервал;
```

Определить *i* – интервал, образованный нотами *n1* и *n2* (*n1*<*n2*): секунда – это интервал из двух соседних нот (например, ре и ми), терция – интервал через ноту (например, фа и ля) и т. д.

7.13*. Язык Паскаль не разрешает вводить и выводить значения перечислимого типа с помощью процедур *read* и *write*. А что делать, если по алгоритму нужны эти операции?

Выписать фрагменты программы для решения каждой из следующих задач.

```
type цвет = (черный, серый, белый);
     буква = (a, b, c);
var x: цвет;  y: буква;
```

а) Вывести на экран значение переменной *x*.

б) Ввести заданное во входном файле значение типа *буква* (т.е. *a*, *b* или *c*) и присвоить его переменной *y*.

7.14. Ответить на следующие вопросы об ограниченном типе, определяемом конструктором типа (диапазоном) α..β.

а) Можно ли вставлять пробел между двумя точками в диапазоне α..β?

б) Величины α и β должны быть одного и того же типа данных, который называется базовым (для данного ограниченного типа). Каким может быть этот базовый тип: целым, вещественным, логическим, символьным, перечислимым?

в) Верно ли, что α и β обязательно должны быть константами и не могут быть переменными или более общими выражениями? Какое из неравенств должно выполняться: α<β или α≤β? Допускается ли пустой ограниченный тип (α>β)?

г) Верно ли, что в ограниченный тип входят все величины базового типа, которые больше или равны α и меньше или равны β? Можно ли определить ограниченный тип, составленный из нескольких отрезков базового типа (например, из целых чисел от 1 до 9 и от 15 до 22) или из отдельных величин базового типа (например, из простых чисел)?

д) Переменной ограниченного типа можно присваивать только величины из диапазона, определяющего этот тип. А что будет, если в программе этой переменной присваивается величина, не принадлежащая этому диапазону? Кто должен следить за правильностью присваиваний переменным ограниченного типа – автор программы или компьютер?

е) К переменной ограниченного типа можно применять любые операции, которые можно применять и к переменным базового типа, за исключением присваивания, которое выполняется особым образом. В чем же тогда выгода от ограниченных типов?

ж) Согласно описанию
```
var x: '0'..'9';  y: '0'..'9';
```
типы переменных *x* и *y* различны, т.к. определены разными (хотя внешне и одинаковыми) конструкторами типа, и в то же время допустимо присваивание *x*:=*y*. Как устраняется это противоречие?

7.15*. Имеются описания
```
type digit = 0..9;
var d: digit;  k: 5..15;  n: integer;
```
Ответить на следующие вопросы.

а) Какой тип является базовым для типа *digit*? Какие величины входят в тип *digit*? Допустимы ли следующие операторы присваивания:
1) d:=7; 2) d:=10; 3) d:= –1; 4) d:=1.0; 5) d:='5' ?

б) Допустим ли следующий оператор:
```
if d+n>0.1*exp(d) then d:=abs(n) mod 10 else d:=d mod k ?
```

в) При каких условиях допустимы следующие присваивания:
1) n:=d; 2) d:=n; 3) d:=k–1 ?

г) Безошибочно ли поработает процедура *read*(*d*), если было

введено число –2?

7.16*. Найти и объяснить ошибки в следующем описании:

```
const n=180;  pi=3.14159;
type цифра = '0'..'9';
     буква = a..z;
     угол = –n..n;
     отрезок = 0..n–1;
     период = –pi..pi;
     плюс = '+'..'+';
     неделя = (вс, пн, вт, ср, чт, пт, сб);
     будни = пн..пт;
     выходной = сб..вс;
```

7.17. Ответить на следующие вопросы об операторе варианта

case E of V_1; V_2; …; V_n end,

где каждый вариант V имеет вид: C_1, C_2, …, C_k: S .

а) Каким может быть тип выражения E (селектора): целым, вещественным, логическим, символьным, перечислимым, ограниченным?

б) Верно ли, что все величины C_i, которые указываются в начале вариантов, должны быть того же типа, что и селектор E, и обязательно должны быть константами, но не переменными или более общими выражениями?

в) Можно ли одну и ту же константу C_i указывать несколько раз в одном и том же варианте? А в разных вариантах?

г) Если в каком-то варианте нужно перечислить константы с подряд идущими значениями (например, 1, 2, 3 и 4), то можно ли вместо них указать диапазон (например, 1..4)?

д) Сколько операторов можно указывать в варианте (т.е. вместо S)? Что делать, если в варианте не нужен ни один оператор? А если нужно несколько операторов?

е) Надо ли указывать точку с запятой за последним вариантом (перед *end*)? Будет ли ошибкой, если ее поставить здесь?

ж) Какова семантика оператора варианта, т.е. как он выполняется? Сколько вариантов выполняется – один или несколько? Что произойдет, если ни одна из констант C_i не совпадает со значением селектора: будет зафиксирована ошибка или оператор варианта проработает как пустой оператор?

з) Обязательно ли указывать в виде констант C_i все значения, которые потенциально может принимать селектор? А если заранее известно, что селектор не будет принимать какие-то из этих значений?

7.18*. Определить значения переменных p и d после выполнения следующих операторов:

```
p:=true; d:=1;
case  k mod 10  of
3, 2, 7, 5:  d:=k;
        1:  ;
     4, 8:  begin  p:=false;  d:=2  end;
     9, 6:  begin  p:=false;  d:=3  end
end {of case}
```

если целая переменная k имеет следующее значение:

а) 6; б) 235; в) 71; г) 100

7.19.

```
var u, w: 'a'..'z';
```

Найти и объяснить ошибки в следующем операторе варианта:

```
case u of
  'a'..'w':  w:=succ(u);
  'u', 'y':  u:='g';  w:=u
end
```

7.20*.

```
type сезон = (зима, весна, лето, осень);
     месяц = (янв, фев, мар, апр, май, июн, июл, авг,
              сен, окт, ноя, дек);
var m: месяц;  s: сезон;
```

Определить s – сезон, на который приходится месяц m.

7.21.

```
type страна = (Германия, Куба, Лаос, Монако, Непал, Польша);
     континент = (Азия, Америка, Европа);
var s: страна; c: континент;
```

По s – названию страны определить c – название ее континента.

7.22.

```
type единица = (дециметр, километр, метр,
                миллиметр, сантиметр);
     длина = real;
var x: длина;  p: единица;
```

Значение переменной x, означающее некоторую длину в единицах p, заменить на величину этой же длины в метрах.

7.23.

```
var k: 1..9;
```

Вывести значение переменной k римскими цифрами.

7.24. Для целого числа k от 1 до 99 вывести фразу «мне k лет», учитывая при этом, что при некоторых значениях k слово «лет» надо заменить на слово «год» или «года».

7.25. Для натурального числа k вывести фразу «мы нашли k гри-

бов в лесу», согласовав окончание слова «гриб» с числом k.

7.26. type падеж = (им, род, дат, вин, твор, предл);
слово = (степь, боль, тетрадь, дверь);
var w: слово; p: падеж;

Вывести слово w в падеже p и единственном числе (например, при *w=степь* и *p=твор* надо вывести слово *степью*).

7.27. type курс = (С, В, Ю, З); {север, восток, юг, запад}
приказ = (вперед, вправо, назад, влево);
var К1, К2: курс; ПР: приказ;

Корабль сначала шел по курсу *К1*, а затем его курс был изменен согласно приказу *ПР*. Определить *К2* – новый курс корабля.

7.28*. type месяц = (янв, фев, мар, апр, май, июн, июл, авг, сен, окт, ноя, дек);
var d: 28..31; m: месяц;

Переменной d присвоить количество дней в месяце m (год считать невисокосным).

7.29. var y: 1901..2099; m: месяц; {см. 7.28}
d: 1..31; t: boolean;

Переменной t присвоить значение *true*, если тройка y, m, d (год, месяц, число) образует правильную дату, и значение *false* – иначе (при 31 июня и т. п.).

7.30. var d, d1: 1..31; m, m1: месяц; {см. 7.28}
y: 1901..2099; y1: 1901..2100;

По дате d, m, y (число, месяц, год) определить $d1$, $m1$, $y1$ – дату следующего дня.

7.31. var k: 1..366; d: 1..31; m: месяц; {см. 7.28}

а)* определить k – порядковый номер того дня високосного года, который имеет дату d, m;

б) определить d, m – дату k-го по счету дня високосного года.

7.32. type число = 1..31;
месяц = (янв, фев, мар, апр, май, июн, июл, авг, сен, окт, ноя, дек);
деньнедели = (вс, пн, вт, ср, чт, пт, сб);
var d: число; m: месяц; wd1, wd: деньнедели;
k: 0..12;

Считая, что год невисокосный и его 1 января приходится на день недели $wd1$, определить:

а) wd – день недели, на который приходится день с датой d, m;

б) k – количество понедельников в году, приходящихся на 13-е числа.

7.33. Найти и объяснить все ошибки в каждой из следующих программ.

а)
```
program ошибки (input, output);
  type месяц = (янв, фев, мар, апр, май, июн, июл, авг,
                сен, окт, ноя, дек);
       зима = дек..фев;
       весна = мар..май;
  var m: месяц; k: 1..12;
begin
  read(m);
  if m>весна then m:=июнь;
  for k:=ord(янв) to ord(m) do m:=succ(m);
  writeln(m)
end.
```

б)
```
program ошибки (input, output);
  type цифра = '0' . '9';
       знак = ('+', '–', '*', '/');
  var d: цифра; t: boolean;
begin
  read(d);
  case d of
     2, 3, 5, 7: t:=true; d:=succ(d);
  0, 1, 4, 6, 8, 9: t:=false;
  writeln(t, d)
end.
```

8. РЕГУЛЯРНЫЕ ТИПЫ: ВЕКТОРЫ

8.1. Ответить на следующие вопросы о массивах регулярного типа
array [*ТИ*] of *ТЭ*,
где *ТИ* – тип индекса и *ТЭ* – тип элементов массива.

а) Все элементы массива должны быть одного и того же типа *ТЭ*. Каким может быть этот тип? Например, может ли он быть регулярным, т.е. допускаются ли в языке Паскаль массивы, элементами которых являются массивы?

б) Верно ли, что число элементов в массиве фиксировано и не может меняться в процессе выполнения программы? В какой момент должно стать известным число элементов в массиве – на этапе написания программы или на этапе ее выполнения?

в) С какой целью индексируются элементы массива?

г)* Верно ли, что тип индекса *ТИ* массива может быть любым простым типом языка Паскаль, кроме типа *real*? Какие из следующих конструкций можно использовать в качестве *ТИ*, а какие нет?

1) 1..10; 2) 1..1; 3) –5..5; 4) 0..maxint;
5) integer; 6) real; 7) 'a'..'z'; 8) '9'..'0';
9) char; 10) (a,b,c); 11) (a1,a3,a5); 12) boolean

д) Как по типу индекса *ТИ* определяются число элементов в массиве, индексы первого и последнего элементов массива? Указать это число и эти индексы для следующих *ТИ*:

1) 0..9; 2) 1..maxint; 3) (a,b,c); 4) boolean; 5) char

е) Какие объекты входят в регулярный тип *array*[1..10] *of boolean*?

ж) Можно ли в Паскаль-программе использовать массив, если число элементов в нем заранее (при составлении программы) неизвестно? Почему в конструкторе регулярного типа *array*[1..*k*] *of char* нельзя в качестве верхней границы индекса (*k*) указать переменную? Почему в таком типе рекомендуется описывать верхнюю границу как константу программы?

з) Какие начальные значения получают элементы массива при его описании?

и) Что такое упакованные массивы? Как они описываются? Отличается ли их обработка от обработки неупакованных массивов? Каковы достоинства и недостатки упакованных массивов по сравнению с неупакованными? Кто (автор программы или транслятор) определяет, упаковывать массив или нет?

8.2*. Имеются описания:

```
type день = (вчера, сегодня, завтра);
     вектор = array [1..30] of real;
var a: вектор;
    b: packed array [–2..2] of (x,y,z);
    c: array ['0'..'9'] of вектор;
    d: array [день] of 0..23;
```

Для каждого из массивов *a, b, c* и *d* указать:

а) сколько в нем элементов;

б) какие значения могут принимать его элементы;

в) каковы индексы его первого и последнего элементов.

8.3*. Описать регулярный тип *R*, объединяющий в себе массивы, элементами которых являются натуральные числа, а индексами – любые символы.

8.4*. Найти и объяснить ошибки в следующем разделе описаний:

```
const n=50;
type   слово = packed array [0..n–1] of буква;
       буква = 'a'..'z';
       вектор = array [real] of integer;
       цифры = array [false..true] of (1, 2, 3, 4);
var  k: 1..maxint;
     x: array [1..k] of char;
     y: packed array [–n..n] of 0..0;
     z: array [(a, b, c)] of boolean;
```

8.5. Ответить на следующие вопросы об операциях над массивами.

а) Как в программе сослаться на массив как единое целое?

б) Распространяется ли на регулярные типы правило, что каждый конструктор типа определяет новый тип, отличный от всех других типов?

в)* В языке Паскаль допускается присваивание массивов как единых объектов. Должны ли при этом массивы быть одного и того же типа или они могут быть разных типов?

Пусть, к примеру, имеются описания

```
type M = array [1..10] of real;
var a, b: M;   c: array [1..10] of integer;
    d: array [0..9] of real;   e, f: array [1..10] of real;
```

Какие из следующих операторов присваивания правильные, а какие нет?

1) a:=b; 2) a:=c; 3) b:=d; 4) a:=e; 5) e:=f

г) Какие еще операции, помимо присваивания, можно применять к массивам как единым объектам? Например, можно ли сравнивать массивы, складывать их, вводить и выводить? Что делать, если нужно выполнить какую-то операцию над массивом, а ее нет в языке Паскаль?

д) Что обозначает переменная с индексом (индексированная переменная) $x[i]$, где x – имя массива, а i – индекс? Верно ли, что эта переменная имеет тот же тип, что и элементы массива x, и может использоваться так же, как и любая другая переменная этого типа?

е)* Что можно указывать в качестве индекса в индексирован-

ной переменной: константу (например, число), переменную, выражение? Должно ли значение индекса принадлежать типу индекса массива? А что будет, если указан иной индекс?

Пусть, например, имеются описания

```
var x: array[0..9] of integer;  y: array[boolean] of char;
    k: integer;
```

Какие из следующих индексированных переменных правильные, а какие нет и почему?

1) x[6]; 2) x[10]; 3) x[k mod 10]; 4) x[5.0]; 5) y['a'=y[k>4]]

ж) В чем различие между записями *x1* и *x*[1]?

8.6*.
```
const n=41;
var x: array [1..n] of real;  y: real;
```

Написать фрагмент программы для вычисления:

а) $y = \sqrt[n]{|x_1 \cdot x_2 \cdot \ldots \cdot x_n|}$;

б) $y = \max x_i$;

в) $y = x_1 - x_2 + x_3 - \ldots - x_{n-1} + x_n$;

г) $y = x_1x_n + x_2x_{n-1} + \ldots + x_nx_1$;

д) $y = x_1^2 + x_3^2 + x_5^2 + \ldots + x_n^2$;

е) $y = x_n\,(x_n+x_{n-1})\,(x_n+x_{n-1}+x_{n-2}) \ldots (x_n+ \ldots +x_1)$.

8.7*.
```
var av: array[101..118] of real;  n: 101..118;
```

Пусть *av*[*k*] означает среднюю оценку, полученную на некотором экзамене студентами из группы с номером *k*. Присвоить переменной *n* номер группы, студенты которой лучше всего сдали этот экзамен. (Считать, что такая группа единственная.)

8.8.
```
var d: array[–40..40] of 0..365;  av: real;
```

Пусть *d*[*t*] означает число дней в (невисокосном) году, когда среднесуточная температура в некотором городе была (округленно) равной *t* градусам. Определить *av* – среднегодовую температуру в этом городе.

8.9.
```
var B: array[boolean] of boolean;
```

Каждому элементу массива *B* присвоить значение, равное его индексу.

8.10.
```
var occur: array['0'..'9'] of 0..maxint;  k: –1..1;
```

Пусть *occur*[*d*] означает число вхождений цифры *d* в некоторый текст. Переменной *k* присвоить значение –1, 0 или 1, если общее число вхождений четных цифр в этот текст соответственно меньше, равно или больше общего числа вхождений нечетных цифр.

8.11*.
```
type текст = packed array [1..72] of char;
     шифр = packed array [char] of char;
var t: текст;  k: шифр;
```

Зашифровать текст *t*, заменив в нем каждый символ на значение элемента массива *k*, индексом которого является этот символ.

8.12*.
```
type ДеньНедели = (пн, вт, ср, чт, пт, сб, вс);
var год: array [1..365] of ДеньНедели;
```

Присвоить каждому элементу *год*[*i*] название того дня недели, на который приходится *i*-й по счету день невисокосного года, если известно, что 1 января – среда (*год*[1]=*ср*, *год*[2]=*чт* и т.д.).

8.13.
```
type имя = (Валя, Гена, Женя, Коля, Маша, Нина,
            Саша, Таня, Федя, Шура);
var Пол:  array [имя] of (муж, жен);
    Рост: array [имя] of 140..200;
    И: имя;  Ср: real;
```

По массивам *Пол* и *Рост* определить:

а)* *И* – имя самого рослого мужчины;

б) *Ср* – средний рост женщин.

8.14.
```
type месяц = (янв, фев, мар, апр, май, июн, июл, авг,
              сен, окт, ноя, дек);
var КД: array [месяц] of 28..31;
```

Присвоить каждому элементу массива *КД* значение, равное количеству дней в соответствующем месяце (невисокосного) года.

8.15.
```
type страна = (Грузия, Россия, Украина);
     город = (Киев, Москва, Одесса, Сочи,
              Тбилиси, Томск);
var x: array [1..20] of город;
```

Вывести название страны, города которой наиболее часто встречаются в массиве *x* (считать, что такая страна единственная).

8.16.
```
const n=100;
var x, y, z: array[1..n] of integer;  s: integer;  eq: boolean;
```

Реализовать в виде фрагментов программы следующие операции над массивами.

а)* Ввод массива: ввести заданные во входном файле числа x_1, x_2, …, x_n и присвоить их соответствующим элементам массива *x*.

б) Вывод массива: вывести значения всех элементов массива *x*.

в) Сложение массивов: вычислить $z=x+y$.

г) Скалярное произведение: вычислить $s=x_1y_1+x_2y_2+\ldots+x_ny_n$.

д)* Сравнение массивов: переменной *eq* присвоить значение *true*, если массивы *x* и *y* (поэлементно) равны, и значение *false* иначе.

е)* Копирование массива: переписать содержимое массива x в массив y.

8.17. var x: array [1..40] of char; y: packed array [1..40] of char;
Переписать содержимое массива x в массив y. (Можно ли эту задачу решить с помощью оператора присваивания $y:=x$?)

8.18. Программа. Ввести последовательность из 60 целых чисел и вывести ее на экран в обратном порядке по 6 чисел в строке.

8.19*. var x: array[1..10] of integer; p: integer; eq: boolean;
Требуется присвоить переменной *eq* значение *true*, если в массиве x есть элемент, равный p, и значение *false* иначе.

Ниже предложены три решения этой задачи. Какие из них правильные, а какие нет и почему?

а) eq:=true; for i:=1 to 10 do if x[i]=p then goto 1; eq:=false; 1:

б) i:=1; while (i<=10) and (x[i]<>p) do i:=i+1; eq:=i<=10 ;

в) i:=0; repeat i:=i+1 until (x[i]=p) or (i=10); eq:=x[i]=p;

(*Подсказка*: учесть, что в любой операции сначала вычисляются ее операнды и только затем выполняется сама операция.)

8.20. const n=40;
var x: array [1..n] of integer; s: integer; t: boolean;
Написать фрагмент программы для решения следующей задачи:

а) переменной s присвоить сумму элементов массива x, предшествующих первому нулевому элементу, или сумму всех элементов, если нулевого элемента нет;

б) переменной t присвоить значение *true*, если элементы массива x упорядочены строго по возрастанию, и значение *false* иначе;

в) переменной t присвоить значение *true*, если массив x симметричен (т.е. его элементы, равноудаленные от концов, равны), и значение *false* иначе.

8.21*. Указать, для решения каких из следующих задач нужны массивы, а в каких задачах можно обойтись и без них?

а) Дано 50 чисел. Найти их среднее арифметическое.

б) Дано 50 чисел. Определить, сколько среди них отличается от последнего числа.

в) Дано 100 чисел. Вывести сначала все отрицательные из них, а затем все остальные.

г) Дано число a. Определить первый отрицательный член последовательности $x_1, x_2, x_3, \ldots$, где $x_1=a$, $x_n=tg(x_{n-1})$ при $n>1$.

8.22. Программа. Дано целое число b. Вывести величины $x_0, x_1, \ldots, x_{99}$, где:

а) $x_0=b$, $x_n = x_{[n/2]} + x_{n-1}$ при $n\geq 1$ ([α] означает целую часть числа α);

б) $x_0=1$, $x_1=b$, $x_n=x_{n-2}+(-1)^n x_{n-1}$ при $n\geq 2$.

8.23. Программа. Вычислить величину

$$(x_1y_1+x_3y_3+ \ldots +x_{29}y_{29}) / (x_2y_2+x_4y_4+ \ldots +x_{30}y_{30}),$$

если числа для ввода заданы в следующем порядке:

а) $x_1, x_2, \ldots, x_{30}, y_1, y_2, \ldots, y_{30}$;

б) $x_1, y_1, x_2, y_2, \ldots, x_{30}, y_{30}$

В этой задаче первый способ задания исходных данных удобен для пользователя программы, а второй – для программиста. Какому из этих двух способов следует отдать предпочтение и почему?

8.24. Требуется вычислить (по схеме Горнера) значение многочлена

$$a_0x^{20} + a_1x^{19} + \ldots + a_{19}x + a_{20}$$

в точке t, если исходные данные заданы для ввода в следующем порядке:

а) $t, a_0, a_1, \ldots, a_{20}$;

б) $a_0, a_1, \ldots, a_{20}, t$.

Написать программу решения этой задачи, причем так, чтобы программу можно было «перенастроить» на многочлен другой степени минимальными изменениями в ее тексте.

8.25. Программа. Дан непустой текст из цифр, за которым следует точка. Определить:

а)* сколько различных цифр входит в этот текст;

б) сколько раз каждая цифра входит в этот текст;

в) какая цифра чаще других входит в этот текст (если таких цифр несколько, вывести любую из них).

8.26*. Пусть в задаче нужно хранить некоторый набор величин при условии, что точное число этих величин в наборе неизвестно, но известно максимально возможное число их. Можно ли в такой ситуации использовать массив, и если да, то как?

Написать программу, которая вводит текст, содержащий от 1 до n символов (n=70), за которым следует точка, и выводит этот текст в обратном порядке.

8.27. Программа. Дан текст из n символов (n=80). Вывести сначала все цифры, входящие в него, а затем все остальные символы, сохраняя при этом исходное взаимное расположение символов в каждой из этих двух групп.

8.28. Программа. Дана последовательность целых чисел $x_1, x_2, \ldots, x_n$ (n=30). Вывести эту же последовательность, но удалив из нее все вхождения наименьшего из чисел x_i.

8.29. const n=100;
var x: array [1..n] of real;
Не используя вспомогательный массив, преобразовать массив x по следующему правилу:

а) поменять местами максимальный и минимальный элементы массива (считать, что в массиве все элементы различны);

б)* элементы массива расположить в обратном порядке:
$(x_1, x_2, \ldots, x_{n-1}, x_n) \rightarrow (x_n, x_{n-1}, \ldots, x_2, x_1)$;

в) переставить местами левую и правую половины массива;

г)* элементы массива циклически сдвинуть на одну позицию влево:
$(x_1, x_2, x_3, \ldots, x_n) \rightarrow (x_2, x_3, \ldots, x_n, x_1)$;

д) элементы массива циклически сдвинуть на одну позицию вправо;

е) элементы массива циклически сдвинуть на две позиции влево;

ж) ненулевые элементы массива перенести в его начало, а все нулевые – в конец;

з) переставить элементы массива так, чтобы слева от (новой позиции) x_1 оказались меньшие его элементы, а справа – бóльшие или равные.

8.30. var x, y: array [1..70] of real; k: 1..69;
Используя массив y как вспомогательный, преобразовать массив x по следующему правилу:

а) все отрицательные элементы массива x перенести в его начало, а все остальные – в конец, сохраняя исходное взаимное расположение как среди отрицательных, так и среди остальных элементов;

б) элементы массива x циклически сдвинуть на k позиций влево.

8.31. const n=30;
var s: array [1..n] of 1..10000; k: integer;
Подсчитать k – число тех элементов массива s, которые являются:

а)* степенями двойки (1, 2, 4, 8, 16, ...);

б) полными квадратами (1, 4, 9, 16, 25, ...);

в) числами Фибоначчи (1, 2, 3, 5, 8, 13, ...).

8.32. const n=70;
var z: array[1..n] of real; k: 0..n;
Подсчитать k – число тех элементов массива z, которые строго больше всех предшествующих им элементов (считать, что z_1 не относится к таким элементам).

8.33. const n=20;
var s: packed array [1..n] of char;
Вывести на экран следующую таблицу из символов s_i массива s:

$s_1\ s_2\ s_3\ \ldots\ s_{n-1}\ s_n$
$s_2\ s_3\ s_4\ \ldots\ s_n\ s_1$
$\ldots$
$s_n\ s_1\ s_2\ \ldots\ s_{n-2}\ s_{n-1}$

8.34*. type фамилия = (Бетелин, Войскунский, Коннов, Мальковский, Пильщиков, Школьный);
имя = (Василий, Владимир, Игорь, Михаил, Олег, Юрий);
var MM511: array [фамилия] of имя;
ЕстьТезки: boolean;
Переменной *ЕстьТезки* присвоить значение *true*, если в массиве *MM511* есть одинаковые имена, и значение *false* иначе.

8.35. const n=50;
var x: array[1..n] of integer; k: integer;
Подсчитать k – число различных элементов в массиве x.

8.36. var t: array [1..365] of real;
m: (янв, фев, мар, апр, май, июн, июл, авг, сен, окт, ноя, дек);
По массиву t, где указана температура каждого дня некоторого невисокосного года, определить m – название месяца с наибольшей среднемесячной температурой.

8.37*. const k=50; m=20; n=70; {n=k+m}
var x: array [1..k] of real; y: array [1..m] of real;
z: array [1..n] of real;
Элементы каждого из массивов x и y упорядочены по неубыванию. Объединить элементы этих двух массивов в один массив z так, чтобы и он был упорядочен по неубыванию.

8.38*. const n=31;
var x: array [1..n] of integer; p: integer; k: 1..n;
found: boolean;
Пусть элементы массива x упорядочены по возрастанию. Если элемент p входит в массив x, то переменной *found* присвоить значение *true*, а переменной k – номер элемента массива x, равного p, и присвоить переменной *found* значение *false* в противном случае.

Предложены два варианта решения этой задачи методом бинарного (двоичного) поиска:

```
1) a:=1; b:=n;  found:=false;
   repeat
     k:=(a+b) div 2;
     if p=x[k] then found:=true else
      if p>x[k] then a:=k+1 else b:=k–1
   until found or(a>b);

2) a:=1; b:=n;
   repeat
     k:=(a+b) div 2;  if p>x[k] then a:=k+1 else b:=k
   until a>=b;
   found:=x[k]=p;
```

Для каждого из этих вариантов ответить на следующие вопросы.

а) Чему равно число сравнений искомого элемента p с элементами массива x в худшем (для поиска) случае – когда искомого элемента нет в массиве?

б) Если p не найден в x и если p надо было бы вставить в x с сохранением упорядоченности (считая, что в x есть свободная $(n+1)$-я позиция), то как по окончанию поиска определить индекс позиции для вставки p?

8.39. var x: array[1..700] of 1..701; y: 1..701;

Пусть все элементы массива x различны и расположены по возрастанию. Используя метод бинарного поиска, найти y – то единственное целое число из отрезка [1, 701], которого нет в этом массиве.

8.40. const n=1000;
var x: array [1..n] of integer; p: integer;

Пусть первые $n–1$ элемент массива x упорядочены по неубыванию, а n-я позиция этого массива пока свободна. Требуется вставить новый элемент p в этот массив с сохранением упорядоченности по неубыванию. (Для поиска в $x[1..n–1]$ места для p использовать метод бинарного поиска.)

8.41. const n=100;
var x: array [1..n] of real;

Упорядочить массив x по неубыванию (т.е. переставить его элементы так, чтобы для всех k выполнялось $x_k \le x_{k+1}$), используя следующий алгоритм сортировки (упорядочения).

а)* *Сортировка выбором.* Найти максимальный элемент и поменять его местами с последним элементом. Затем применить этот же метод ко всем элементам, кроме последнего (он уже находится на своем окончательном месте), и т.д.

б) *Сортировка обменом* (*метод пузырька*). Последовательно сравнивать пары соседних элементов x_k и x_{k+1} (k=1, 2, 3, ..., n–1) и, если $x_k > x_{k+1}$, переставлять их; тем самым наибольший элемент окажется на своем месте в конце массива. Затем применить этот же метод ко всем элементам, кроме последнего, и т.д. (Учесть, что если при очередном просмотре массива не было сделано ни одной перестановки, то массив уже упорядочен.)

в) *Сортировка простыми вставками.* Пусть первые k элементов массива уже упорядочены по неубыванию. Сравнивая $(k+1)$-й элемент с предыдущими элементами (в обратном направлении), найти его место среди этих элементов и вставить его сюда, предварительно освободив это место сдвигом части предыдущих элементов на одну позицию вправо; тем самым будут упорядочены уже $k+1$ первых элементов массива. Повторять эти действия для всех k от 1 до n–1.

г) *Сортировка бинарными вставками.* Аналогична сортировке простыми вставками, но место для $(k+1)$-го элемента среди предыдущих элементов определяется бинарным поиском.

8.42. type страна = (Алжир, Греция, Дания, Индия, Китай, Перу, Россия, Франция, Япония);
var S: array[1..8] of страна;

Упорядочить элементы массива S по алфавиту.

8.43. Программа. Дано пятизначное целое число. Составить из его цифр максимально возможное число. (Например, 25702 → 75220.)

8.44. type слово = packed array [1..10] of char;
var x, y: слово; ag: boolean;

Переменной *ag* присвоить значение *true*, если слово y является анаграммой слова x, т.е. отличается от него только порядком расположения символов, и значение *false* иначе, считая при этом, что:

а) в каждом из слов x и y нет повторяющихся символов;

б) в словах x и y могут быть повторяющиеся символы.

8.45. var k: 10000..99999;
d: packed array [1..5] of '0'..'9';

Выписать фрагмент программы для решения следующей задачи:

а) в массив d записать (слева направо) цифры целого числа k;

б) получить k – целое, составленное (слева направо) из цифр массива d.

8.46.
```
const n=80;
type цифра = 0..9;
     число = packed array [1..n] of цифра;
var a, b, c: число; carry: boolean;
```
Рассматривая a и b как последовательности цифр (слева направо) десятичной записи некоторых неотрицательных целых чисел, получить c – аналогичное представление для суммы этих двух чисел. Если в сумме окажется более n значащих цифр, то ее левую цифру отбросить, а переменной *carry* присвоить значение *true*, иначе переменной *carry* присвоить значение *false*. (Операции умножения и деления не использовать.)

8.47.
```
type мантисса = packed array [1..9] of 0..9;
     порядок = packed array [1..2] of 0..9;
var m: мантисса; p: порядок; x: real;
```
Выписать фрагмент программы для решения следующей задачи:

а) вещественное число $0.m_1m_2...m_9 \cdot 10^{p_1p_2}$, составленное из цифр массивов m и p, присвоить переменной x

б) представив вещественное число x ($x \geq 0.1$) в виде $0.m_1m_2... \cdot 10^{p_1p_2}$ ($m_1 \neq 0$), записать в массив m первые 9 цифр его мантиссы, а в массив p – обе цифры его десятичного порядка.

8.48.
```
type мантисса = packed array [1..30] of 0..9;
     порядок = packed array [1..2] of 0..9;
var mx, my, mz: мантисса;
    px, py, mz: порядок; err: boolean;
```
Пусть в массиве mx хранятся старшие 30 цифр мантиссы, а в массиве px – две цифры десятичного порядка некоторого вещественного числа $0.m_1m_2... \cdot 10^{p_1p_2}$ ($m_1 \neq 0$) и пусть в массивах my и py аналогичным образом представлено другое вещественное число (оба этих числа ≥ 0.1). Получить в массивах mz и pz аналогичное представление для суммы этих двух чисел, сохраняя в сумме только старшие 30 цифр мантиссы, первая из которых должна быть отлична от 0. В случае переполнения суммы (показатель ее порядка больше 99) переменной *err* присвоить значение *true*, а иначе – значение *false*. (Операции умножения и деления не использовать.)

8.49.
```
type элемент = 0..99;
     множество = packed array [элемент] of boolean;
var x, y, z: множество; t: boolean;
```


Рассматривая массивы x, y и z как представления некоторых множеств из величин типа *элемент* ($x[k]$=*true*, если элемент k принадлежит множеству x, и $x[k]$=*false* иначе, и т.п.), реализовать следующие операции над этими массивами-множествами:

а) переменной t присвоить значение *true*, если множество x является подмножеством множества y, и значение *false* иначе;

б) $z = x \cap y$ – пересечение множеств;

в) $z = x \cup y$ – объединение множеств;

г) $z = x \setminus y$ – разность множеств (в z входят все элементы из x, которые не входят в y).

8.50.
```
const n=20; n1=21; {n1=n+1}
var P, Q: array [0..n] of real; R: array [0..n1] of real; a: real;
```
По P – массиву коэффициентов многочлена

$$P(x) = p_0x^n + p_1x^{n-1} + \ldots + p_{n-1}x + p_n$$

получить:

а)* R – массив коэффициентов многочлена $(x-a)P(x)$;

б) Q – массив коэффициентов многочлена $P(x+a)$.

8.51. Программа. Дана последовательность из n различных целых чисел (n=100). Найти сумму чисел, расположенных между максимальным и минимальным элементами этой последовательности (в сумму включить и оба этих элемента).

8.52. Программа. Дано n вещественных чисел (n=40). Определить порядковый номер того из них, который наиболее близок к среднему арифметическому всех этих чисел (считая, что такой элемент единственный).

8.53. Программа. Даны координаты n точек на плоскости: x_1, y_1, ..., x_n, y_n (n=20). Найти номера двух точек, расстояние между которыми наибольшее (считать, что такая пара точек единственная).

8.54. Программа. Даны две последовательности по n целых чисел в каждой (n=30). Найти наименьшее среди тех чисел первой последовательности, которые не входят во вторую последовательность (считая, что хотя бы одно такое число есть).

8.55. Программа. Дана последовательность из n целых чисел (n=20). Определить количество инверсий в этой последовательности (т.е. таких пар элементов, в которых большее число находится слева от меньшего: $x_i > x_j$ при $i<j$).

8.56. Программа. Дан текст из малых латинских букв, за которым следует точка. Вывести в алфавитном порядке все буквы, которые

входят в этот текст по одному разу.

8.57. Программа. Ввести заданный текст из n символов (n=100) и вывести его, удалив повторные вхождения каждого символа.

8.58. Программа. Определить, сколько различных символов входит в заданный текст, содержащий не более n символов (n=100) и оканчивающийся точкой (в сам текст точка не входит).

8.59. Программа. Дана непустая последовательность слов из малых латинских букв; между соседними словами – запятая, за последним словом – точка. Вывести в алфавитном порядке все буквы, которые входят в наибольшее количество слов этой последовательности.

8.60. Программа. Подсчитать количество «счастливых» шестизначных автобусных билетов, т.е. таких, в номерах которых сумма трех первых цифр равна сумме трех последних. (Воспользоваться тем, что число «счастливых» билетов равно $s_0^2+s_1^2+\ldots+s_{27}^2$, где s_n – количество чисел от 0 до 999, сумма цифр которых равна n.)

9. РЕГУЛЯРНЫЕ ТИПЫ: МАТРИЦЫ

9.1. Ответить на следующие вопросы о массиве A:

```
var A: array [1..15] of array [0..33] of real;
```

а) Верно ли, что в языке Паскаль матрицы (прямоугольные двухмерные таблицы) рассматриваются как частный случай массивов – как массив массивов?

б) Эквивалентны ли следующие конструкторы регулярных типов:
array [1..15] of array [0..33] of real и array [1..15, 0..33] of real ?

в) Что означает переменная $A[5]$? Каков тип этой переменной? Можно ли строки матрицы обрабатывать как единые объекты? Допустимо ли присваивание $A[5]:=A[8]$ и что оно означает? Какие еще операции можно применять к строкам матрицы как единым объектам? Например, можно ли их сравнивать, вводить или выводить?

г) Какие операции можно применять к столбцам матрицы как единым объектам?

д) Что означает переменная $A[5][22]$? Можно ли эту переменную записать как $A[5,22]$? Что здесь обозначает первый индекс 5 – номер строки или номер столбца матрицы? А что обозначает второй индекс 22? Каков тип переменной $A[5,22]$? Какие операции можно применять в этой переменной?

9.2*. Программа. Ввести квадратную вещественную матрицу n-го порядка (n=4), элементы которой заданы для ввода построчно, и вывести ее на экран по столбцам.

9.3*.
```
const n=20;
var A, B, C: array [1..n, 1..n] of real;
    x, y: array [1..n] of real;
```
Реализовать в виде фрагментов программы следующие операции:
а) $C=A+B$; б) $y=Ax$; в) $C=A\cdot B$; г) $B=B^T$ (транспонировать).

9.4*.
```
var C, D: array [0..9, –5..3] of integer; t: boolean;
```
Переменной t присвоить значение *true*, если матрицы C и D равны, и значение *false* иначе.

9.5.
```
const n=50;
var B: array [1..n, 1..n] of integer; m, r, c: integer; d: boolean;
```
Предполагая, что все элементы матрицы B различны, решить указанную задачу.

а) Найти m – значение наибольшего элемента матрицы B.

б)* Найти m – минимум среди элементов матрицы B, расположенных ниже главной диагонали.

в)* Найти номер строки (r) и номер столбца (c), в которых находится наибольший элемент матрицы B.

г) Переменной d присвоить значение *true*, если наименьший элемент матрицы B находится выше главной диагонали.

д) Поменять местами наибольший и наименьший элементы матрицы B.

9.6*.
```
type точка = array [(x,y)] of real;
var M: array [1..40] of точка; d: real;
```
Рассматривая элементы массива M как координаты точек на плоскости, найти d – наибольшее расстояние между этими точками.

9.7.
```
type остров = (Барбадос, Гаити, Гренада, Куба,
               Мартиника, Ямайка);
     месяц = (янв, фев, мар, апр, май, июн, июл, авг,
              сен, окт, ноя, дек);
var t: array [остров, месяц] of real;
    i: остров; m: месяц;
```
Элемент $t[x,y]$ означает среднемесячную температуру на острове x в месяце y. Определить, какой месяц (m) и на каком острове (i) самый холодный.

9.8.
```
type страна = (Алжир, Египет, Заир, Камерун, Конго,
```

Ливия, Мали, Нигер, Судан, Чад, Эфиопия);
var соседи: array [страна, страна] of boolean;
с: страна;

Элемент *соседи[a,b]* равен *true*, если страны *a* и *b* имеют общую границу, и равен *false* иначе (*соседи[a,a]=false*). Определить *с* – страну, имеющую общие границы с наибольшим числом стран.

9.9*. type M = array[char, boolean] of integer;
var X: M; Y: M; Z: array[char, boolean] of integer;

Одинаковы ли типы строк матриц *X* и *Y* и матриц *X* и *Z*? Какие из следующих присваиваний допустимы, а какие нет и почему?

а) X['0']:=Y['*']; б) X['0']:=Z['*']; в) X['0', true]:=Z['*', false]

9.10. type вектор = array [1..20] of integer;
матрица = array [1..20] of вектор;
var A: матрица; x: вектор;
B: array [1..20, 1..20] of integer;

Выписать фрагмент программы для решения следующей задачи:

а)* нечетные строки матрицы *A* заменить на *x*;

б)* четные столбцы матрицы *A* заменить на *x*;

в)* первые шесть строк массива *B* заменить на *x*;

г) в матрице *A* поменять местами 1-ю и 2-ю строки, 3-ю и 4-ю строки, ..., 19-ю и 20-ю строки (воспользоваться массивом *x* как вспомогательным);

д) аналогичным образом поменять местами строки в массиве *B*.

9.11. var A: array[1..7, 1..4] of real;

Переставляя строки и столбцы матрицы *A*, добиться того, чтобы ее наибольший элемент (один из них) оказался в верхнем левом углу.

9.12. var A: packed array [1..20, 1..20] of boolean;
B: packed array [1..19, 1..19] of boolean;
n, k: 1..20;

Получить массив *B* из массива *A* удалением *n*-й строки и *k*-го столбца.

9.13. Программа. Дана вещественная матрица размером *n*×*m* (*n*=20, *m*=6). Упорядочить ее строки по неубыванию:

а)* их первых элементов;

б) суммы их элементов;

в) их наибольших элементов.

9.14. var A: array [1..10, 1..10] of integer;

Заполнить массив *A* следующим образом:

а)* $\begin{pmatrix} 0&0&0&\dots&0\\ 0&1&0&\dots&0\\ 0&0&2&\dots&0\\ \dots&&&&\\ 0&0&0&\dots&9 \end{pmatrix}$ б) $\begin{pmatrix} 1&2&\dots&10\\ 11&12&\dots&20\\ 21&22&\dots&30\\ \dots&&&\\ 91&92&\dots&100 \end{pmatrix}$ в) $\begin{pmatrix} 1&2&3&\dots&10\\ 0&1&2&\dots&9\\ 0&0&1&\dots&8\\ \dots&&&&\\ 0&0&0&\dots&1 \end{pmatrix}$

9.15. var A: array [1..6, 1..9] of real; x: real;

Заполнить массив *A* по следующему правилу: $A_{i,j}=x^i/j!$

9.16. type месяц = (янв, фев, мар, апр, май, июн, июл, авг, сен, окт, ноя, дек);
дн = (пн, вт, ср, чт, пт, сб, вс, нет);
календарь = array [месяц, 1..31] of дн;
var K: календарь;

Заполнить календарь *K* соответствующими днями недели (для несуществующих дат указать *нет*) при условии, что год невисокосный и 1 января – понедельник (*K*[*янв*,1]=*пн*; *K*[*янв*,2]:=*вт*; ...; *K*[*фев*,29]:=*нет*; ...).

9.17*. var A: array [1..9, 1..9] of real; s: real;

Найти *s* – сумму элементов из заштрихованной области массива *A* (см. рис. 3).

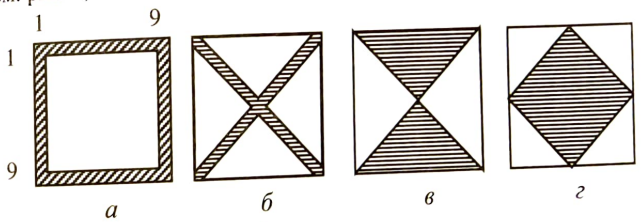


Рис. 3

9.18. var A: array [1..15, 1..20] of integer;
b: array [1..15] of boolean;

По массиву *A* получить массив *b*, присвоив его *k*-му элементу значение *true*, если выполнено указанное условие, и значение *false* иначе:

а)* все элементы *k*-й строки массива *A* нулевые;

б) элементы *k*-й строки массива *A* упорядочены по убыванию;

в) *k*-я строка массива *A* симметрична;

г) в *k*-й строке массива *A* есть одинаковые элементы.

9.19. const n=8; m=12;
var k: integer; C: array [1..n, 1..m] of integer;

Определить *k* – количество «особых» элементов массива *C*, считая

элемент «особым», если:

а) он больше суммы остальных элементов своего столбца;

б) в его строке слева от него находятся элементы, меньшие его, а справа – бóльшие;

в) он меньше ближайших соседей в своей строке и в своем столбце.

9.20. var A: array [1..10, 1..15] of char; c: char; OK: boolean;

Переменной *OK* присвоить значение *true*, если в матрице *A* имеется символ *c*, и значение *false* иначе.

9.21. var k: integer;
C: array [1..13, 1..18] of char;

Определить *k* – количество различных элементов массива *C* (т.е. повторяющиеся элементы считать один раз).

9.22. Имеется таблица *T* результатов некоторого шахматного турнира, в котором участвовало *n* шахматистов (*n*>2):

T: array [1..n, 1..n] of (В, Н, П, Х),

где *T*[*i*,*j*]=*В*, если *i*-й участник выиграл у *j*-го (при этом *T*[*j*,*i*]=*П*), *T*[*i*,*j*]=*Н*, если *i*-й и *j*-й участники сыграли вничью, и *T*[*i*,*i*]=*Х*. Возможный вид таблица (при *n*=3):

$$\begin{pmatrix} X & B & П \\ П & X & H \\ B & H & X \end{pmatrix}$$

За выигрыш дается 1 очко, за ничью – 0.5 очка, за проигрыш – 0 очков. Вывести на экран номера участников в порядке невозрастания набранных ими очков.

9.23. type судья = 1..9;
фамилия = (Абрамов, Большакова, Бордаченкова, Волкова, Вылиток, Головин, Горячая, Грацианова, Кацкова, Мальковский, Матвеева, Пильщиков, Родин, Руденко, Трифонов);
var баллы: array[фамилия, судья] of 0..10;
итоги: array[1..15] of фамилия;

Перечисленные в типе *фамилия* люди участвовали в некотором соревновании, которое оценивали 9 судей: *баллы*[*m*,*r*] – балл (от 0 до 10), выставленный участнику *m* судьей *r*. Итоги соревнования подводились по следующей схеме: из всех баллов, полученных участником, отбрасывались крайние (наибольший и наименьший) баллы, а

остальные суммировались, что и давало итоговую оценку участника; чем больше оценка у участника, тем выше его место.

Записать в массив *итоги* фамилии участников в порядке занятых ими мест (при равенстве итоговых оценок участников располагать в алфавитном порядке).

9.24. type имя = (Анна, …, Юрий);
родство = (сын, дочь, отец, мать, нет);
var TP: array [имя, имя] of родство;
И, В, Д: имя; k: integer;

На основе (непротиворечивой) таблицы родства *TP* (*TP*[*x*,*y*]=*нет*, если *y* не является ни родителем, ни ребенком *x*'а, *TP*[*x*,*y*]=*сын*, если *y* – сын *x*'а, и т.п.) присвоить переменной:

а)* *В* – имя одной (любой) из внучек человека с именем *И*, если такие есть;

б) *Д* – имя одного (любого) из дядей человека с именем *И*, если такие есть (дядями считать только братьев родителей);

в) *k* – число всех кузин и кузенов (двоюродных сестер и братьев) человека с именем *И*.

9.25. const n=256;
type screen = packed array [1..n, 1..n] of 0..1;
var S: screen;

Преобразовать массив *S*, осуществив поворот элементов вокруг его центра на 90° против часовой стрелки.

9.26. Программа. Даны натуральное *n* и квадратная вещественная матрица *m*-го порядка (*m*=5). Вычислить *n*-ю степень этой матрицы (A^1=*A*, A^2=*A*·*A*, A^3=A^2·*A* и т.д.).

9.27. Программа. Определить, является ли заданная квадратная целая матрица *n*-го порядка (*n*=10) симметричной относительно главной диагонали.

9.28. Программа. Элемент матрицы назовем седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце.

Для заданной целой матрицы размером *n*×*m* (*n*=10, *m*=15) вывести на экран индексы всех ее седловых точек.

9.29. Программа. Дана квадратная вещественная матрица *n*-го порядка (*n*=7), все элементы которой различны. Найти скалярное произведение строки, в которой находится наибольший элемент матри-

цы, на столбец с наименьшим элементом.

9.30. Программа. Определить, является ли заданная квадратная целая матрица n-го порядка (n=10) ортонормированной, т.е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1.

9.31. Программа. Определить, является ли заданная квадратная целая матрица n-го порядка (n=9) порядка магическим квадратом, т. е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.

9.32. Программа. Дана непустая последовательность слов из малых латинских букв; слова разделяются запятыми, за последним словом – точка. Среди всех пар a_i и b_i, где a_i – первая, а b_i – последняя буквы i-го слова последовательности, определить наиболее часто встречающуюся пару.

9.33. Программа. По заданным коэффициентам a_{11}, a_{12}, …, a_{1n}, a_{22}, a_{23}, …, a_{2n}, …, a_{nn} (n=20, $a_{ii}\neq 0$) и правым частям b_1, b_2, …, b_n найти решение «треугольной» системы линейных уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ \ldots\ + a_{1n}x_n = b_1 \\ a_{22}x_2 + a_{23}x_3 + \ \ldots\ + a_{2n}x_n = b_2 \\ a_{33}x_3 + \ \ldots\ + a_{3n}x_n = b_3 \\ \qquad\qquad \ldots \\ a_{nn}x_n = b_n \end{cases}$$

10. РЕГУЛЯРНЫЕ ТИПЫ: СТРОКИ

10.1*. Дать определение строковых типов в языке Паскаль и указать, какие из следующих типов являются строковыми, а какие нет и почему.

```
const n=20;
type  a = array [1..30] of char;
      b = packed array [1..n] of char,
      c = packed array [0..n] of char;
      d = packed array ['1'..'9'] of char;
      e = packed array [1..n, 1..n] of char;
      f = packed array [1..10] of 'a'..'z';
      g = packed array [1..1] of char;
```


10.2.
```
const k=4;
var S: packed array[1..k] of char;
    T: packed array[1..4] of char;
    U: array [1..4] of char;
    V: packed array [1..2] of char;
```

Ответить на следующие вопросы об операциях над строками.

а) Верно ли, что к строкам (массивам строковых типов) применимы любые операции, которые применимы и к массивам других типов? Как, например, осуществляется доступ к i-му элементу строки S?

б) Верно ли, что все строки равной длины относятся к одному и тому же строковому типу, даже если при их описании использовались разные конструкторы типа?

в)* Верно ли, что допускаются присваивания любых строк равной длины? А разной длины? Какие из следующих присваиваний правильные, а какие нет и почему?

1) S:=T; 2) S:=U; 3) T:=U; 4) S:=V

г) Как записываются строки-константы? Как записать кавычку в строке-константе?

д)* Относится ли строка-константа '1234' к тому же строковому типу, что и строки S и T? Какие из следующих присваиваний правильные, а какие нет и почему?

1) S:='1234'; 2) T:='1234'; 3) U:='1234'; 4) V:='1234'

е) Если всем элементам массива S надо присвоить значение '*', то можно ли это сделать одним оператором присваивания? Если да, то как? А как такое же значение присвоить всем элементам массива U?

ж) Что означает *'a'* – строку из одного символа или величину типа *char*?

з)* Верно ли, что можно сравнивать любые строки равной длины? А разной длины? Что такое лексикографическая упорядоченность строк? Когда две строки равны и когда одна строка меньше другой?

Пусть, к примеру, все элементы массивов S, U и V имеют одно и то же значение *'a'*. Для каждого из следующих отношений указать, допустимо ли оно, и, если да, определить его значение:

1) S='aaba'; 2) S<'aaba'; 3) S=U; 4) S>=V

и) Можно ли строки выводить с помощью процедуры *write*? А вводить с помощью процедуры *read*?

10.3*. Программа. Ввести последовательность из 60 символов и вывести ее на экран дважды, в двух строках.

10.4*. var a, b, c: packed array[1..20] of char;

Используя c как вспомогательный массив, перераспределить строки a и b так, чтобы в a оказалась меньшая из них, а в b – другая.

10.5.
```
type слово = packed array [1..5] of char;
     список = array [1..60] of слово;
var C: список;
```
Вывести на экран:

а)* все слова из списка C, отличные от слова *'hello'*;

б) то слово из списка C, которое лексикографически предшествует всем остальным словам этого списка (считать, что все слова различны);

в)* текст, составленный из последних символов всех слов списка C;

г) все слова из списка C, содержащие ровно две буквы *'d'*.

10.6*.
```
type набор = array [1..8] of char;
     слово = packed array [1..8] of char;
var a1: array [1..20] of набор;
    a2: array [1..20] of слово;
    k1, k2: integer;
```
Подсчитать $k1$ – количество наборов массива $a1$, совпадающих с его первым набором, и $k2$ – количество слов массива $a2$, совпадающих с его первым словом.

10.7*.
```
var s: packed array [1..6] of char;
```
Во входном файле *input* задано от 1 до 6 букв, за которыми следует точка. Ввести эти буквы и записать их в начало строки s, дополнив конец строки пробелами.

10.8. В языке Паскаль можно сравнивать только строки равной длины. А что делать, если требуется сравнить строки разной длины?

Написать программу, которая вводит два слова, в каждом из которых от 1 до 8 малых латинских букв и за каждым из которых – пробел, и которая выводит эти слова в алфавитном порядке. (Считать, что в типе *char* пробел предшествует любой букве.)

10.9*.
```
type color = (red, blue, green, yellow, black, white);
var x: color;
```
Во входном файле задана последовательность малых латинских букв, за которой следует пробел. Если это правильная запись значения типа *color*, то присвоить его переменной x, иначе сообщить об ошибке.

10.10*.
```
const v = 'aeiou';
type строка = packed array [1..200] of char;
var s: строка; k: 0..200;
```


Определить, сколько раз (k) в строку s входят символы, перечисленные в константе v. (Допустима ли запись v[1]?)

10.11. Найти и объяснить ошибки в следующей программе:
```
program errors (input, output);
  const sign = '+–*/';
  var s: packed array [1..10] of char;
      i, j, k: integer;
begin
  read(s);
  for i:=1 to 10 do
    for j:=1 to 4 do
      if s[i]=sign[j] then k:=k+1;
  if k<6 then writeln(s) else writeln(sign)
end.
```

10.12. В типе *char* русские буквы (малые и большие) необязательно упорядочены по алфавиту. Как определить, является ли некоторый символ русской буквой и, если да, какой ее порядковый номер в русском алфавите?

Написать программу для решения следующей задачи.

а)* Ввести текст из 60 символов и вывести на экран все малые русские буквы, входящие в этот текст.

б) Дан непустой текст из больших русских букв, за которым следует точка. Определить, упорядочены ли буквы этого текста по алфавиту.

в) Ввести текст из малых русских букв, за которым следует точка, и вывести на экран этот же текст, но большими русскими буквами.

10.13. Программа. Вывести в алфавитном порядке все различные малые русские буквы, входящие в заданный текст из n символов (n=200).

10.14. Программа. Ввести целое число от 1 до 39 (за числом – пробел) и вывести его, записав римскими цифрами. (*Рекомендация*: использовать массив с записью римскими цифрами чисел от 1 до 9.)

10.15.
```
type строка = packed array [1..80] of char;
     var s: строка;
```
Пусть в начале строки s находится не более 40 латинских букв, а за ними следуют пробелы. Преобразовать эту строку следующим образом:

а) все вхождения *'abc'* заменить на *'def'*;

б)* удалить первое вхождение буквы *'w'*, если такое есть;

в) удалить все вхождения *'th'*;

г)* заменить на *'ks'* первое вхождение *'x'*, если такое есть;

д) после каждой буквы *'q'* добавить букву *'u'*;

е) заменить все вхождения *'ph'* на *'f'*, а все вхождения *'ed'* на *'ing'*.

10.16. Программа. Дана последовательность, содержащая от 1 до *n* слов (*n*=30), в каждом из которых от 1 до *k* малых латинских букв (*k*=5); между соседними словами – запятая, за последним словом – точка. Вывести на экран:

а) эту же последовательность слов, но в обратном порядке;

б) те слова, перед которыми в последовательности находятся только меньшие (по алфавиту) слова, а за ними – только бóльшие;

в) эту же последовательность слов, но удалив из нее повторные вхождения слов;

г) все слова, которые встречаются в последовательности по одному разу;

д) все различные слова, указав для каждого из них число его вхождений в последовательность;

е) все слова в алфавитном порядке.

10.17. Программа. Дана последовательность, содержащая от 2 до *n* слов (*n*=50), в каждом из которых от 1 до *k* малых латинских букв (*k*=8); между соседними словами – не менее одного пробела, за последним словом – точка. Вывести те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству:

а) слово симметрично;

б) первая буква слова входит в него еще раз;

в) буквы слова упорядочены по алфавиту;

г) слово совпадает с начальным отрезком латинского алфавита (*a*, *ab*, *abc* и т.д.); учесть, что в диапазоне *'a'..'z'* могут быть символы, отличные от латинских букв;

д) слово совпадает с конечным отрезком латинского алфавита (*z*, *yz*, *xyz* и т.д.);

е) длина слова максимальна;

ж) в слове нет повторяющихся букв;

з) каждая буква входит в слово не менее двух раз;

и) в слове гласные буквы (*a*, *e*, *i*, *o*, *u*) чередуются с согласными.

10.18. Программа. Дана последовательность, содержащая от 2 до *n* слов (*n*=30), в каждом из которых от 2 до *k* латинских букв (*k*=10); между соседними словами – не менее одного пробела, за последним словом – точка. Вывести все слова, отличные от последнего слова, предварительно преобразовав каждое из них по следующему правилу:

а) перенести первую букву в конец слова;

б) перенести последнюю букву в начало слова;

в) удалить из слова первую букву;

г) удалить из слова последнюю букву;

д) удалить из слова все последующие вхождения первой буквы;

е) удалить из слова все предыдущие вхождения последней буквы;

ж) оставить в слове только первые вхождения каждой буквы;

з) если слово нечетной длины, то удалить его среднюю букву.

10.19. Программа. Дан текст из *n* символов (*n*=60). Вывести этот текст, подчеркивая (ставя минусы в соответствующих позициях следующей строки экрана) все входящие в него большие и малые русские буквы.

10.20. Программа. Дана последовательность, содержащая от 1 до *n* слов (*n*=90), в каждом из которых от 1 до *k* малых русских букв (*k*=10); между соседними словами – не менее одного пробела, за последним словом – точка. Вывести эти слова по алфавиту.

10.21. Программа. Даны два символа – латинская буква (от *'a'* до *'h'*) и цифра (от 1 до 8), например *a2* или *g5*. Рассматривая их как координаты поля шахматной доски, на котором находится ферзь, нарисовать на экране шахматную доску, пометив крестиками все поля, которые «бьет» этот ферзь, и ноликами все остальные поля.

10.22. Решить предыдущую задачу для шахматного коня.

11. ПРОЦЕДУРЫ И ФУНКЦИИ

11.1. Считая, что процедура *P* описывает решение некоторой подзадачи *S*, ответить на следующие вопросы.

а) Пусть в программе приходится несколько раз решать одну и ту же подзадачу *S*. Если решение этой подзадачи описать в виде процедуры *P*, то в чем выгода от этого? Что проще и короче – каждый раз явно выписывать решение этой подзадачи или каждый раз выписывать обращение к процедуре *P*?

б) При использовании процедур приходится описывать их и обращаться к ним. Как называется раздел программы, где описываются процедуры? Надо ли в этом разделе описывать стандартные процедуры (*read*, *write* и т.п.)? Как называется обращение к процедуре? Относится ли оно к выражениям или к операторам?

в) Описание процедуры состоит из заголовка и блока. Какая информация о процедуре указывается в заголовке? Верно ли, что блок процедуры аналогичен блоку всей программы? Каково назначение этого блока?

г) Верно ли, что параметрами процедуры P называются величины, от которых зависит решение подзадачи S, т.е. это исходные данные и результаты данной подзадачи? Например, каковы параметры процедуры, которая по корням приведенного квадратного трехчлена находит его коэффициенты?

д) Параметры процедуры P приходится указывать дважды – при ее описании и при обращении к ней. Верно ли, что параметры, которые используются при описании процедуры, называются формальными параметрами и что они обозначают исходные данные и результаты подзадачи S «в общем виде»? Верно ли, что параметры, которые указываются при обращении к процедуре, называются фактическими параметрами и что они обозначают конкретные данные, при которых требуется решить подзадачу S?

е) Важен ли порядок перечисления формальных параметров в заголовке процедуры? Влияет ли он на порядок следования фактических параметров при обращении к процедуре?

ж) В заголовке процедуры надо указать имя и тип каждого формального параметра. Есть ли ограничения на имена формальных параметров, могут ли они, например, совпадать с именами переменных, констант и других объектов программы? Любого ли типа может быть параметр процедуры? Верно ли, что этот тип обязательно должен указываться по имени, но не с помощью конструктора типа? Например, какие формальные параметры неправильно описаны в следующем заголовке:

procedure P (a: real; b: array[1..10] of char; a: boolean) ?

з) Параметры процедур делятся на параметры-значения и на параметры-переменные. Верно ли, что исходные данные подзадачи S следует описывать в процедуре P как параметры-значения, а ее результаты – как параметры-переменные? Чем отличается описание параметра-значения от описания параметра-переменной? Например, если имеется заголовок

procedure P (a: real; var b: char; c: boolean)

то какие из параметров этой процедуры являются параметрами-значениями, а какие – параметрами-переменными?

и) Один и тот же параметр может быть одновременно и исходным данным подзадачи, и ее результатом, как, например, параметр

процедуры $cube(x)$, реализующей действие $x:=x^3$. Как следует описывать этот параметр – как параметр-значение или как параметр-переменная?

к) Основным считается случай, когда в заголовке процедуры каждый формальный параметр описывается независимо от других. Однако заголовок можно сократить, если несколько параметров имеют одинаковый тип. Например, как более коротко записать заголовок

procedure P (a: real; b: real; var c: char; var d: char) ?

Для любых ли однотипных параметров допускаются подобные сокращения? Например, эквивалентны ли следующие пары заголовков:

1) procedure P (a:real; b:char; c:real) и procedure P (a,c:real; b:char);

2) procedure P (var a:real; b:real) и procedure P (var a,b:real) ?

л) В описании процедуры за заголовком следует блок, состоящий из раздела описаний и составного оператора, называемого телом процедуры. Верно ли, что тело описывает алгоритм решения соответствующей подзадачи в общем виде? Можно ли в этом алгоритме использовать вспомогательные объекты (переменные, константы и т.д.)? Где они описываются? Могут ли имена этих вспомогательных объектов совпадать с именами формальных параметров процедуры или с именами других объектов программы?

м) Обращение к процедуре P (оператор процедуры) – это требование решить подзадачу S при конкретных данных, указываемых фактическими параметрами. Через запятую или через точку с запятой должны перечисляться фактические параметры в операторе процедуры? Должны ли число, порядок следования и типы фактических параметров совпадать с числом, порядком следования и типами формальных параметров этой процедуры? Если формальный параметр описан как параметр-значение, то что можно указать в качестве фактического параметра: константу, переменную или любое выражение соответствующего типа? Верно ли, что если формальный параметр описан как параметр-переменная, то в качестве фактического параметра может быть задана только переменная соответствующего типа? Чем объясняется это ограничение?

Пусть, к примеру, имеются описания

```
var a, b: integer;  c: char;
procedure P (var x: integer; y: integer); …
```

Какие из следующих операторов процедуры неправильные и почему?

1) P(a); 2) P(c,b); 3) P(a,5); 4) P(a,5.0);
5) P(a,b); 6) P(a,2*b+1); 7) P(1,b); 8) P(2*a,2*b)

н) Используя правило языка Паскаль о том, что каждый конструктор типа определяет новый тип, отличный от всех других типов, объяснить на следующем примере, почему в заголовке процедуры тип формального можно указывать только по имени:

```
var x: (a,b);
procedure P (var y: (a,b)); …
  …
P(x);
  …
```

Как можно исправить ошибку в этом фрагменте программы?

о) Можно ли в языке Паскаль описать процедуру без параметров? Как записывается ее заголовок? Как выглядит обращение к ней? Привести пример обращения к какой-нибудь стандартной процедуре без параметров.

11.2. Решение каждой из указанных подзадач описать в виде процедуры, ответив предварительно на следующие вопросы: Сколько параметров у этой процедуры? Каких они типов? Какие из них должны быть параметрами-значениями, а какие – параметрами-переменными?

Используя эту процедуру, выписать фрагмент программы для решения указанной далее задачи.

а)* var c, d: integer;

Подзадача: привести заданную дробь к несократимому виду.

Задача: вычислить величину $1+\frac{1}{2}+\frac{1}{3}+\ \ldots\ +\frac{1}{20}$ в виде несократимой дроби $\frac{c}{d}$.

б) var k: integer; t: boolean;

Подзадача: найти наибольшую и наименьшую цифры в записи неотрицательного целого числа.

Задача: изменить значение логической переменной t на противоположное, если у чисел $k+1$ и k^2 ($k\geq 0$) совпадают и наибольшие, и наименьшие цифры.

в)* var a, b, c: real;

Подзадача: значения двух вещественных переменных перераспределить так, чтобы в первой из них оказалось большее значение, а во второй – меньшее.

Задача: значения переменных a, b и c перераспределить так, чтобы стало $a\geq b\geq c$.

г) type M = array [1..20] of real;
var x, y: M;

Подзадача: в массиве типа M заменить наибольший элемент на заданную величину.

Задача: заменить на 0 максимальный элемент в том из массивов x и y, где первый элемент меньше (считать, что в этих массивах первые элементы не равны, а максимальные элементы единственные).

д) type строка = packed array [1..10] of char;

Подзадача: упорядочить символы строки по алфавиту.

Задача: логической переменной t присвоить значение *true*, если строка x является анаграммой строки y (т.е. если x отличается от y только порядком символов), и значение *false* иначе.

е)* *Подзадача*: вывести на экран строку из 13 минусов.

Задача: вывести на экран следующую таблицу:

```
-------------
   a    not a
-------------
  true false
 false  true
-------------
```

11.3*. Программа. Даны произвольные вещественные числа a, b, c и d. Для каждой тройки этих чисел, для которой можно построить треугольник с такими длинами сторон, вывести на экран площадь данного треугольника.

(*Рекомендация*: описать процедуру вывода площади треугольника для одной тройки чисел.)

11.4. Программа. Даны два слова, в каждом из которых от 1 до 8 строчных латинских букв и за каждым из которых следует пробел. Вывести эти слова в алфавитном порядке.

(*Рекомендация*: описать процедуру ввода одного слова и дополнения его пробелами до 8 символов.)

11.5. Программа. Вычислить сумму трех заданных квадратных вещественных матриц n-го порядка (n=6) и вывести ее.

(*Рекомендация*: описать процедуру ввода одной матрицы.)

11.6. Ответить на следующие вопросы о передаче параметров по значению.

а) Верно ли, что при вызове процедуры (по оператору проце-

дуры) вычисляется каждый фактический параметр-значение и его значение передается процедуре, которая создает новую переменную с именем и типом соответствующего формального параметра и присваивает ей данное значение, после чего в своем теле использует эту переменную вместо формального параметра, а по окончанию своей работы уничтожает ее?

б)* Почему результат подзадачи, решаемой процедурой, нельзя описывать как параметр-значение? Пусть, к примеру, имеются описания

```
var a: integer;
procedure P (x: integer);
 begin write(x); x:=x+1; write(' ', x) end;
```

Что будет выдано на экран при выполнении следующих операторов:

```
a:=2; P(a); write(' ', a) ?
```

Почему процедура *P* не изменила значение фактического параметра *a*?

в) Можно ли формальный параметр-значение использовать в теле процедуры как дополнительную переменную, если, конечно, ее начальное значение (значение фактического параметра) уже использовано и далее не нужно? Например, правильно ли процедура

```
procedure P (x: real; var y: real);
 begin x:=x*x; y:=x+1/x end;
```

реализует присваивание $y=x^2+x^{-2}$?

г) Обязательно ли тип формального параметра-значения должен в точности совпадать с типом соответствующего фактического параметра? Например, если имеется описание

```
procedure P (x: real); begin write(x:1:5) end;
```

то допустимо ли обращение *P*(7)? Каким в общем случае должно быть соотношение между типом формального параметра-значения и типом соответствующего фактического параметра?

11.7. Ответить на следующие вопросы о передаче параметров по ссылке.

а) Верно ли, что при вызове процедуры (по оператору процедуры) ей передается ссылка на каждый фактический параметр-переменную (т.е. адрес места в памяти компьютера, занимаемое этой переменной) и что при выполнении тела процедуры все обращения к соответствующему формальному параметру означают обращения к этой фактической переменной?

б)* Сохраняются ли присваивания параметру-переменной, сделанные в процедуре, по выходу из процедуры? Если, например, имеются описания

```
var a: integer;
procedure P (var x: integer);
 begin write(x); x:=x+1; write(' ', x) end;
```

то что будет выдано на экран при выполнении следующих операторов:

```
a:=2; P(a); write(' ', a) ?
```

Обязательно ли результат подзадачи, решаемой процедурой, должен описываться как параметр-переменная?

в) Можно ли параметру-переменной ничего не присваивать в процедуре? Когда есть выгода от этого, а когда ее нет?

Пусть имеются следующие два варианта описания процедуры, которая выводит на экран начальный и конечный элементы массива, в первом из которых этот массив передается по значению, а во втором – по ссылке:

```
type M = array [1..200] of integer;
var A: M;
procedure Pr1 (X: M); begin write(X[1], ' ', X[200]) end;
procedure Pr2 (var X: M); begin write(X[1], ' ', X[200]) end;
```

Объяснить на примере обращений *Pr1*(*A*) и *Pr2*(*A*), почему параметры сложных типов (например, регулярных) невыгодно (с точки зрения памяти и времени) описывать как параметры-значения и почему рекомендуется описывать их как параметры-переменные, даже если в процедуре этим параметрам ничего не присваивается?

Почему эта рекомендация не подходит для параметров простых типов? Пусть, к примеру, процедура *делит*(*a*), которая выводит на экран делители натурального числа *a*, имеет заголовок

```
procedure делит (var a: integer)
```

Если *k* – целая переменная, то допустимы ли обращения *делит*(*k*), *делит*(15) и *делит*(2**k*+1)? Как правильно обратиться к этой процедуре в двух последних случаях? Удобны ли такие обращения?

г) Обязательно ли тип формального параметра-переменной должен в точности совпадать с типом соответствующего фактического параметра? Если такого совпадения типов не требовать, то что произойдет при обращениях *P*(*k*) и *Q*(*a*), где:

```
var k: integer; a: real;
procedure P (var x: real); begin x:=x+1.2 end;
procedure Q (var x: integer); begin x:=x mod 10 end;
```

д) Пусть имеются описания

```
var a: integer;
```

```
procedure P (var x: integer);
  var a: integer;
begin x:=0 end;
```

Какой переменной с именем a – из программы или вспомогательная из процедуры – будет присвоено значение 0 в результате обращения $P(a)$? Ответ обосновать.

е)* Если в качестве фактического параметра-переменной указана переменная с индексом (например, $x[i]$), то как происходит передача этого параметра: либо сначала вычисляется индекс и тем самым определяется конкретный элемент массива, а затем в процедуру передается ссылка на этот конкретный элемент, либо в процедуру передается ссылка на массив, а индекс будет вычисляться уже в теле процедуры?

Определить, к примеру, что выведет на экран следующая программа:

```
program print (output);
  var i: integer; x: array[1..2] of integer;
  procedure P (var a: integer); begin i:=2; a:=0 end;
begin
  x[1]:=1; x[2]:=2; i:=1; P(x[i]); writeln(x[1], ' ', x[2])
end.
```

ж) В качестве фактического параметра-переменной нельзя указывать элемент упакованного массива. Объяснить этот запрет, опираясь на следующие два факта: 1) в памяти компьютера адреса имеют ячейки, но не их части; 2) в ячейках памяти размещается по несколько соседних элементов упакованного массива.

11.8. Что выдаст на экран следующая программа?

а)
```
program print (output);
  var a, b: integer;
  procedure exchange (x: integer; var y: integer);
    var z: integer;
    begin z:=x; x:=y; y:=z end;
begin
  a:=1; b:=2; exchange(a, b); writeln(a, ' ', b)
end.
```

б)
```
program print (output);
  var k, m, n: integer;
  procedure NOD (a, b: integer; var c: integer);
    begin
      while a<>b do if a>b then a:=a–b else b:=b–a;
      c:=a
    end;
begin
  k:=10; m:=15; NOD(k,m,n); writeln(k, ' ', m, ' ', n)
end.
```


в)
```
program print (output);
  var k, m, n: integer;
  procedure NOD (var a, b, c: integer);
    begin
      while a<>b do if a>b then a:=a–b else b:=b–a;
      c:=a
    end;
begin
  k:=10; m:=15; NOD(k,m,n); writeln(k, ' ', m, ' ', n)
end.
```

11.9. Найти и объяснить ошибки в следующей программе.

а)
```
program errors (output);
  const a='1234';
  type string = packed array [1..4] of char;
  var b: string;
  procedure P (s1: string; var s2: string);
    begin writeln(sl, ' ', s2) end;
begin P('abcd', 'efgh'); P(a,a); b:=a; P(b,b) end.
```

б)
```
program error (output);
  var x: packed array [1..3] of char;
  procedure Q (c: char; var d: char);
    begin d:=succ(c) end;
begin x:='295'; Q(x[1],x[3]); writeln(x) end.
```

11.10*.
```
type вектор = array[1..20] of real;
  var a, b, c, d: вектор;
```

Описать процедуру нахождения суммы двух векторов и, используя ее, вычислить $d=a+b+c$.

11.11.
```
const n=10;
  type матрица = array[1..n, 1..n] of real;
  var A, B, C, D, E: матрица;
```

Описать процедуру умножения двух матриц и, используя ее, вычис-

лить $E=A\cdot B+C\cdot D$.

11.12. Ответить на следующие вопросы о функциях.

а) В чем основное отличие функций от процедур? Можно ли (условно) рассматривать функцию как процедуру со значением?

б) В каком разделе программы описываются функции? Надо ли здесь описывать стандартные функции (*sin*, *trunc*, *ord* и т.п.)? В каком порядке здесь должны описываться функции и процедуры?

в) Верно ли, что у функций такие же правила описания формальных параметров, задания фактических параметров и передачи параметров, что и у процедур?

г) Существуют три отличия описания функции от описания процедуры. Указать их на примере следующего описания функции:

```
function f (x: real): real;
  var y: real;
begin y:=x*x; f:=y+1/y end;
```

д) Какого типа может быть значение функции? Допускаются ли в языке Паскаль векторные, матричные, строковые и другие функции регулярных типов? Верно ли, что в заголовке функции ее тип должен указываться по имени, но не с помощью конструктора типа? Допустим ли, например, следующий заголовок функции:

```
function f (k: integer): 0..9 ?
```

е)* Каков смысл оператора присваивания *f*:=*e*, где *f* – имя описываемой функции, а *e* – выражение? Какого типа должно быть выражение *e*? Сколько таких операторов присваивания может быть в теле функции *f*? Должен ли хотя бы один из них выполниться при вычислении функции *f*? Что будет, если ни одни из них не выполнится? А что будет, если выполнится несколько таких операторов?

Вычислить, к примеру, *f*(1), *f*(2) и *f*(3) при следующем описании функции *f*:

```
function f (k: integer): integer;
 begin if k<3 then  begin f:=1;  if k>1 then f:=2 end  end;
```

ж) В следующем описании функции есть две ошибки:

```
function f (c: char): char;
 begin f(c):=ord(c) end;
```

Что это за ошибки?

з) Можно ли в теле функции использовать ее имя как дополнительную переменную? Есть ли, например, ошибки в следующем описании функции:

```
function f (x: real): real;
```



```
 begin  f:=x;  if f<0 then f:=f+1 end;
```

и, если есть, то что это за ошибки?

и) Верно ли, что обращение к функции называется указателем функции? Относится ли оно к выражениям или к операторам? Верно ли, что внешне (синтаксически) обращения к функции и к процедуре не отличаются друг от друга? Чем должны быть $A(x,y)$ и $B(x,y)$ – функциями и/или процедурами, чтобы был правильным следующий оператор:

```
if A(x,y) then B(x,y) else x:=y ?
```

к) Допускаются ли в языке Паскаль функции без параметров? Как записывается их заголовок? Как выглядит обращение к ним?

л) Используя правило языка Паскаль о том, что каждый конструктор типа определяет новый тип, отличный от всех других типов, объяснить на следующем примере, почему в заголовке функции ее тип можно указывать только по имени:

```
var x: (a,b);
function f (k: integer): (a,b);
 begin  if k=1 then f:=a else f:=b  end;
 …
x:=f(2);
 …
```

Как можно исправить ошибку в этом фрагменте программы?

11.13. var x, y, z: real;

Вычислить $z=(\text{sign } x+\text{sign } y)\cdot\text{sign}(x+y)$, где

$$\text{sign } a=\begin{cases}-1 & \text{при } a<0\\ 0 & \text{при } a=0\\ 1 & \text{при } a>0\end{cases}$$

При решении этой задачи:

а) не использовать функцию *sign*;

б)* определить и использовать функцию *sign*.

11.14.
```
type страна = (Австрия, Гана, Италия, Перу,
                США, Швеция);
     континент = (Америка, Африка, Европа);
var x, y: страна;  t: boolean;
```

Описать функцию *конт*(*s*), определяющую континент, на котором находится страна *s*, и использовать ее для изменения значения переменной *t* на противоположное, если страны *x* и *y* находятся на разных континентах.

11.15*.
```
const n=20;
type вектор = array[1..n] of real;
var a, b, c: вектор;  t: boolean;
```
Описать функцию *скал*(*x*,*y*), вычисляющую скалярное произведение векторов *x* и *y*, и, используя ее, присвоить переменной *t* значение *true*, если векторы *a*, *b* и *c* попарно ортогональны, и значение *false* иначе.

11.16.
```
type натур = 1..maxint;
var a, b: real;  k: натур;
```
Описать функцию *степень*(*x*,*n*) от вещественного *x* и натурального *n*, вычисляющую (через умножение) величину x^n, и использовать ее для вычисления $b=2.7^k+(a+1)^{-5}$.

11.17*. Программа. По заданному вещественному числу *x* вычислить величину

$$\mathrm{sh}(x)\cdot\mathrm{tg}(x+1)-\mathrm{tg}^2(2+\mathrm{sh}(x-1))$$

(*Пояснение*: гиперболический синус *sh x* вычисляется по формуле $(e^x-e^{-x})/2$.)

11.18. Программа. Даны три натуральных числа. Определить их наибольший общий делитель. (*Рекомендация*: описать функцию, вычисляющую наибольший общий делитель двух чисел.)

11.19. Программа. Найти сумму кубов всех трех корней уравнения

$$ex^3-\pi x^2-(2e+1)x+2\pi=0$$

(π=3.14159, e=2.718282), вычисленных с точностью 0.0001. (*Рекомендации*: вручную определить три отрезка, на которых надо искать корни; описать и использовать функцию, которая находит один корень на заданном отрезке.)

11.20*.
```
function f: integer;
  var k: integer;
begin read(k); f:=k end;
```
Вычислить величину *f+f+f*, если во входном файле записаны числа 1, 2, 3 и 4.

11.21. Как следует описывать решение подзадачи – в виде функции или в виде процедуры – в каждом из указанных случаев? Свой выбор обосновать.

а) У подзадачи несколько результатов (например, требуется найти максимальный и минимальный элементы массива).

б) У подзадачи нет результатов (например, требуется вывести массив).

в) У подзадачи один результат, и он сложного типа (например, требуется найти сумму двух матриц).

г) У подзадачи один результат, и он простого типа (например, требуется найти сумму элементов массива).

11.22. Для каждой из следующих задач ответить на вопросы: Какую подзадачу приходится решать в ней несколько раз? Какие исходные данные и результаты у этой подзадачи? Каков их смысл и каких они типов? Как следует описать решение этой подзадачи – в виде функции или в виде процедуры?

Описать данную функцию или процедуру и с ее помощью решить задачу.

а)
```
type строка = packed array [1..60] of char;
  var s, t: строка;  k: integer;
```
Если в первой половине строки *s* менее 12 цифр и если в последней четверти строки *t* нет символов от '*a*' до '*z*', тогда вычислить *k* – количество символов '*', входящих в среднюю треть строки *s*.

б)
```
var a, b: real;  t: boolean;
```
Переменной *t* присвоить значение *true*, если уравнения $x^2+6.2x+a^2=0$ и $x^2+ax+b-1=0$ имеют вещественные корни и при этом оба корня первого уравнения лежат между корнями второго уравнения, и присвоить значение *false* во всех остальных случаях.

в)
```
var a, b, c, m1, m2, m3: real;
```
Рассматривая *a*, *b* и *c* как длины сторон треугольника, найти *m1*, *m2* и *m3* – медианы треугольника, сторонами которого являются медианы исходного треугольника. (*Замечание*: длина медианы, проведенной к стороне *a*, равна $0.5\sqrt{2b^2+2c^2-a^2}$.)

г)
```
type натур = 1..maxint;
  var k: натур;
```
Для каждого из чисел *k*, 2*k*+1 и k^2+1 вывести на экран все его делители.

д)
```
type неотр = 0..maxint;
  var a, b, c, k: неотр;
```
Определить количество палиндромов (т.е. читающихся одинаково слева направо и справа налево) среди чисел *a*, *b* и *c* и присвоить это количество переменной *k*.

е)
```
const n=30;
type массив = array[1..n] of real;
  var x, y: массив;
```

Поменять местами минимальные элементы массивов x и y (считать, что в каждом массиве такой элемент единственный).

ж) type вектор = array [1..40] of integer;
var x, y, z: вектор;

Если наибольший элемент вектора x равен 10 и находится в первой половине этого вектора, и если в векторе y нет положительных элементов, тогда все элементы вектора z, предшествующие его наибольшему элементу, заменить на их кубы. (Считать, что в каждом векторе наибольший элемент один.)

11.23*. Программа. По заданным вещественным числам a_0, a_1, ..., a_{30}, b_0, b_1, ..., b_{30}, c_0, c_1, ..., c_{30}, x, y, z вычислить величину

$$\frac{(a_0x^{30}+a_1x^{29}+\ldots+a_{30})^2-(b_0y^{30}+b_1y^{29}+\ldots+b_{30})}{c_0(x+z)^{30}+c_1(x+z)^{29}+\ldots+c_{30}}$$

11.24. Программа. По заданным 20-элементным целым массивам x и y вычислить

$$u=\begin{cases}\sum\limits_{i=1}^{20}x_i^2 & \text{при } \sum\limits_{i=1}^{15}x_iy_i>0\\ \sum\limits_{i=10}^{20}y_i^2 & \text{иначе}\end{cases}$$

11.25. Программа. По заданным n-элементным вещественным массивам a, b и c (n=50) вычислить

$$t=\begin{cases}\dfrac{\min(b_i)}{\max(a_i)}+\dfrac{\max(c_i)}{\min(b_i+c_i)} & \text{при } \min(a_i)<\max(b_i)\\ \max(b_i+c_i)+\min(c_i) & \text{иначе}\end{cases}$$

11.26. Программа. Даны n-элементные вещественные векторы x, y и z (n=30). Вычислить величину (a,a)–(b,c), где a обозначает тот из этих векторов, в котором самый большой минимальный элемент (считать, что такой вектор единственный), b и c обозначают два других вектора, а (p,q) – скалярное произведение p и q.

11.27. Программа. Даны две квадратные вещественные матрицы n-го порядка (n=10). Вывести на экран квадрат той из них, в которой наименьший след (сумма диагональных элементов), считая, что такая матрица одна.

11.28. Найти и объяснить ошибки в следующей программе:

```
program errors (input, output);
const n=40;
type vector = array [1..n] of real;
var a, b: vector;
function sum (var x, y: vector): vector;
   var i: integer;  z: vector;
  begin for i:=1 to n do z[i]:=x[i]+y[i];  sum:=z end;
procedure reverse (x: vector);
   var i: integer;  r: real;
  begin
    for i:=1 to n div 2 do
      begin r:=x[i]; x[i]:=x[n+1–i];  x[n+1–i]:=r end
  end;
begin
  read(a);  b:=sum(a,reverse(a));  writeln(b)
end.
```

11.29. Внутри указанной процедуры или функции описать и использовать подходящий вспомогательный массив.

а)* const n=500;
type вектор = array[1..n] of 0..99;

Описать функцию *разл*(X), подсчитывающую количество различных элементов в векторе X.

б) const n=100;
type слово = packed array[1..n] of 'a'..'z';

Описать символьную функцию *част*(W), который определяет символ (один из них), который чаще других встречается в слове W.

в) const n=15; m=20;
type матрица = array [1..n, 1..m] of real;

Описать функцию *сум*(A), вычисляющую величину

$x_1x_n+x_2x_{n-1}+\ \ldots\ +x_nx_1$,

где x_i – максимальный элемент i-й строки матрицы A.

г) type сдвиг = 1..19;
шкала = packed array [1..20] of boolean;

Описать процедуру *сдв*(s,k), которая преобразует шкалу s, циклически сдвигая ее элементы на k позиций влево, где k – параметр типа *сдвиг*.

д) type массив = array[1..55] of real;

Описать процедуру *преобр*(X), которая преобразует массив X следующим образом: все отрицательные элементы переносит в его начало, а все остальные – в конец, сохраняя исходный порядок как

среди отрицательных, так и среди остальных элементов.

11.30. Внутри указанной процедуры или функции описать и использовать подходящую вспомогательную процедуру или функцию.

а)* type неотриц = 0..maxint;

Описать вещественную функцию $F(m,n)=n!\cdot m!/(n+m)!$, где n и m – неотрицательные целые числа.

б) type натур = 1..maxint;

Описать логическую функцию *меньше*(a,b) от натуральных чисел, которая принимает значение *true*, если у числа a меньше делителей, чем у числа b, и значение *false* иначе.

в) type вектор = array [1..20] of char;

Описать процедуру *преобр*(x,y,a,b) от четырех векторов, которая преобразует векторы x и y к следующему виду:

$x = (a_1, a_2, \ldots, a_8, x_9, x_{10}, \ldots, x_{20})$,

$y = (y_1, \ldots, y_5, b_1, \ldots, b_6, y_{12}, \ldots, y_{16}, a_1, \ldots, a_4)$.

г) const n=8; m=13;

type матрица = array [1..n, 1..m] of real;

Описать процедуру *swap*(A,B), меняющую местами максимальные элементы матриц A и B. (Считать, что в каждой матрице только один максимальный элемент.)

д) const n=20;

type слово = array[1..n] of 'a'..'z';

Описать процедуру *упор*(a,b,c), которая упорядочивает слова a, b и c по алфавиту: $a \le b \le c$.

11.31. Ответить на следующие вопросы о принципе локализации.

а) Программа, описания процедур и функций состоят из заголовков и блоков, поэтому в программе может быть несколько блоков. Как определяется понятие вложенности одного блока в другой? Если, к примеру, в программе A описаны процедуры B и C, а в процедуре B описана функция D (см. рис. 4), то какие из этих блоков вложены в другие, а какие нет?

б) В каждом блоке могут быть описаны имена (переменных, типов и т.д.) и меток. (Можно считать, что в процедурах и функциях описания начинаются с описания формальных параметров.) Верно ли, что имя (метка), описанное в блоке, локализуется в нем, т.е. этим именем можно пользоваться в данном блоке, но нельзя пользоваться вне блока? Например, если имя u описано в блоке B (см. рис. 4), то можно ли его использовать в блоке D? А в блоке A или C?

в) Верно ли, что одно и то же имя (метку) нельзя дважды описывать в одном блоке, но можно описывать в разных блоках? Можно ли имя x описать и в блоке A, и в блоке B, и в блоке C (см. рис. 4)? Имеет ли оно в этих блоках один и тот же смысл или разный?

г) Пусть имя x описано в блоке A как константа, а в блоке B – как переменная, и пусть блок B вложен в блок A (см. рис. 4). Последнее означает, что, находясь внутри блока B, мы одновременно находимся и внутри блока A. Какое же из двух описаний имени x действует в блоке B, т.е. что здесь обозначает это имя – константу или переменную?

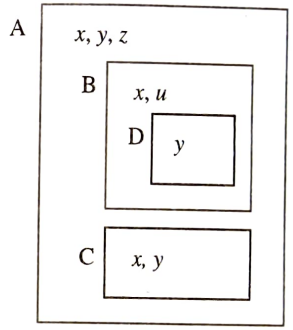


Рис. 4

д) Областью видимости (действия) имени (метки) называется та часть текста программы, где это имя можно использовать и где оно имеет один и тот же смысл. Верно ли, что областью видимости имени (метки) является весь блок, где оно описано, за исключением тех вложенных блоков, в которых описано такое же имя?

Пусть в блоке A (см. рис. 4) описаны имена x, y и z, в блоке B – имена x и u, в блоке C – имена x и y, в блоке D – имя y. Какова область видимости каждого из этих имен?

е) Считается, что любая Паскаль-программа как бы вложена в некоторый суперблок, в котором описаны все стандартные имена (*true*, *sin*, *read* и т.д.). Можно ли описывать такие же имена и в других блоках программы, в частности, в самой программе? Какова область видимости стандартного имени?

Определить, к примеру, правильна ли следующая программа. Если да, что она выведет на экран, а если нет, то в чем причина ошибки?

```
program P (output);
  const t=true; true=false;
  begin writeln(ord(t)<ord(true)) end.
```

ж) Можно ли в блоке *B* (см. рис. 4) использовать имя *z*, описанное в объемлющем блоке *A*? Верно ли, что такое имя называется глобальным по отношению к блоку *B*? Доступно ли в блоке *B* имя *x*, описанное в блоке *A*? Какими именами из блока *A* можно пользоваться в блоке *B*, а какими нельзя?

11.32*. Перечислить локальные и глобальные имена в следующей процедуре:

```
procedure P (x: vect; var y: integer);
  const z='*';
  var c: index;
begin y:=0; for c:=a to b do if x[c]>z then y:=y+1 end;
```

11.33. Что выведет на экран следующая программа?

а)*
```
program print (output);
  var x, y: char;
  procedure P (x: integer);
    const y=true;
    begin writeln(x, ' ', y) end;
  procedure Q;
    var x: char;
    begin x:=succ(y); y:='*'; writeln(x, ' ', y) end;
  begin
    x:='a'; y:='5';
    P(8); writeln(x, ' ', y);
    Q; writeln(x, ' ', y)
  end.
```

б)
```
program print (output);
  type string = packed array [1..5] of char;
  var i: integer; t: string;
  procedure P (var s: string);
    begin
      i:=1; while s[i]<'9' do begin s[i]:=succ(s[i]); i:=i+1 end
    end;
  begin i:=1; t:='12945'; P(t); writeln(t[i]) end.
```

в)
```
program print (output);
  var a, b, c, d: integer;
  procedure P (var b: integer; c: integer);
    var d: integer;
  begin
    a:=5; b:=6; c:=7; d:=8; writeln(a, ' ', b, ' ', c, ' ', d)
  end;
begin
  a:=1; b:=2; c:=3; d:=4; P(a,b); writeln(a, ' ', b, ' ', c, ' ', d)
end.
```

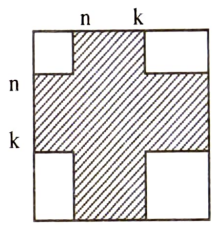



Рис. 5

11.34. type индекс = 1..20;
матрица = array [индекс, индекс] of real;

Описать функцию *max*(*A*,*n*,*k*), где *A* – матрица, а *n* и *k* – индексы (*n*<*k*), вычисляющую наибольший элемент заштрихованной области матрицы *A* (см. рис. 5).

(*Рекомендация*: внутри этой функции описать и использовать вспомогательную функцию, которая находит максимум в одной «прямоугольной» области и для которой матрица *A* является глобальной.)

11.35. type T1 = array [1..10, 1..10] of integer;
T2 = array [1..20, 1..30] of integer;

Описать процедуру *constr*(*A*,*B*,*C*,*D*), которая по матрицам *A*, *B* и *C* типа *T1* строит следующую матрицу *D* типа *T2* (*N* – нулевая матрица типа *T1*):

$$D = \begin{pmatrix} A & B & C \\ B & N & A \end{pmatrix}$$

(*Рекомендация*: внутри этой процедуры описать и использовать вспомогательную процедуру, которая копирует одну матрицу типа *T1* в указанное место матрицы *D* и для которой матрица *D* является глобальной.)

11.36. Программа. Дана непустая последовательность слов, в ка-

ждом из которых от 1 до 6 латинских букв; между соседними словами – запятая, за последним словом – точка. Вывести все слова, у которых одинаковые «соседи», т.е. совпадают предыдущее и следующее слова.

(*Рекомендация*: определить процедуру *readword(w)*, которая вводит очередное слово и присваивает его 6-символьной строке *w*, а запятую или точку, следующую за словом, присваивает некоторой глобальной переменной.)

11.37. Пусть в программе имеются следующие описания:

```
const true=0; succ=1; real=2;
var b: boolean; c, d: char;
```

Ответить (с обоснованием) на следующие вопросы.

а) Можно ли в этой программе логической переменной *b* присвоить значение «истина» или это значение стало недоступным?

б) Можно ли в этой программе переменной *c* присвоить символ, следующий в типе *char* за символом *d*?

в) Можно ли в этой программе использовать вещественные переменные?

11.38. Найти и объяснить ошибки в следующем описании процедуры:

```
procedure errors (var x: boolean);
  const char=0; case=1;
  type a = (true, false); b = ('a','b');
  var c: char;
begin if x then x:=(ord(true)=char) and false end;
```

11.39. Ответить на следующие вопросы о побочных эффектах функций.

а) Что называется побочным эффектом функции?

б) Если функция что-то вводит или выводит, то является ли это побочным эффектом?

в) Если функция меняет значение переменной, описанной вне функции, то является ли это побочным эффектом? Может ли функция произвести такое изменение через свой параметр?

11.40*. Определить, что выдаст на экран следующая программа (считать, что операнды операций вычисляются слева направо):

```
program sideeffect (output);
  var a, b: integer;
  function f (x: integer): integer;
    begin f:=x; a:=0 end;
  function g (var x: integer): integer;
    begin g:=x; x:=0 end;
begin
  a:=1; write(a+f(a),' '); a:=1; write(f(a)+a,' ');
  b:=2; writeln(g(b)=g(b))
end.
```


11.41. Привести такие описания имен *y* и *b*, при которых каждая из процедур *write(y=y)*, *write(y+y=2*y)* и *write(b and b = b)* будет выводить значение *false*.

11.42*. Описать функцию *next* без параметров, которая считывает из входного файла первый символ, отличный от пробела, и объявляет его своим значением. Использовать эту функцию для подсчета *k* – количества отличных от пробела символов текста, который задан во входном файле и за которым следует точка.

11.43. Описать целую функцию *NextInt* без параметров (с побочным эффектом), которая при каждом обращении к ней выдает очередное целое число начиная с 1.

11.44.

```
const d=100; m=5;
  type строка = packed array [1..d] of char;
       подстрока = packed array [1..m] of char;
       позиция = 1..d;
  var x: строка; y, z: подстрока;
```

Описать логическую функцию *поиск(s,ss,k,n)*, проверяющую, входит ли подстрока *ss* в ту часть строки *s*, которая начинается с *k*-й позиции, и, если входит, присваивающую параметру *n* номер позиции, с которой начинается первое вхождение *ss* в эту часть строки *s*.

Используя данную функцию, заменить в строке *x* все вхождения подстроки *y* на подстроку *z*.

11.45. Ответить на следующие вопросы о параметрах-функциях и параметрах-процедурах.

а) Верно ли, что для описания формального параметра-функции (параметра-процедуры) надо выписать полный заголовок функции (процедуры)? Верно ли, что имя функции, указанное в этом заголовке, является именем данного формального параметра и что именно это имя надо указывать при ссылке на данный параметр? Верно ли, что в этом заголовке имена параметров функции могут быть любыми и могут даже повторяться?

б) Что собственно является параметром-функцией – имя функции или обращение к функции? Что надо указывать в качестве соот-

ветствующего фактического параметра – имя конкретной функции или указатель функции? Как передается параметр-функция – по значению или по ссылке?

в) Верно ли, что как фактический параметр-функция (параметр-процедура) нельзя указывать имя стандартной функции (процедуры)? А что делать, если в качестве такого параметра надо использовать именно стандартную функцию?

11.46*. Программа. По заданным вещественным числам c и d ($c<d$) вычислить

$$\int_c^d arctg\, x\,dx + \int_0^{\pi} \sin e^{10x}\,dx$$

Интегралы вычислять приближенно по формуле трапеций при n=20 для первого интеграла и при n=100 для второго:

$$\int_a^b f(x)\,dx \approx h\cdot\left(\frac{f(a)}{2}+\sum_{i=1}^{n-1} f(a+ih)+\frac{f(b)}{2}\right),\ \text{где}\ h=\frac{b-a}{n}$$

11.47. Программа. По заданным 40-элементным вещественным векторам x, y и z вычислить

$$w=\begin{cases}\prod_i(\sin(x_i)+2) & \text{при}\ \prod_i(1-y_i^2)>0.5\\ \prod_i(1-z_i^2) & \text{иначе}\end{cases}$$

11.48. Пусть функция $f(x)$ непрерывна на отрезке $[a,b]$, имеет разные знаки на его концах и имеет только один корень на этом отрезке. Требуется найти этот корень x с точностью eps>0.

Ниже приведены два варианта решения этой задачи методом деления отрезка пополам. Правильно ли они решают эту задачу?

а)
```
repeat
  c:=(a+b)/2;  if f(a)*f(c)<0 then b:=c else a:=c
until b-a<eps;
x:=(a+b)/2
```

б)
```
repeat
  c:=(a+b)/2;  if f(a)*f(c)>0 then a:=c else b:=c
until b-a<eps;
x:=(a+b)/2
```

(*Подсказка*: рассмотреть функцию $f(x)=x$ на отрезке $[-1,1]$.)

11.49. Программа. Вывести в порядке возрастания все 4 корня

уравнений

$$1/(1+x^2)=x,\quad 3e^x+x=0\quad \text{и}\quad x\cdot\ln(1+x)=0.5,$$

вычисленные с заданной точностью ε>0. (*Замечание*: отрезки для поиска корней определить вручную.)

11.50. Программа. Даны координаты вершин двух треугольников. Определить, какой из них имеет бóльшую площадь.

11.51. Программа. Три прямые на плоскости заданы уравнениями $a_kx+b_ky=c_k$ (k = 1, 2, 3). Если эти прямые попарно пересекаются и образуют треугольник, тогда найти его площадь.

11.52. Программа. Даны длины сторон двух треугольников. Определить, являются ли эти треугольники одновременно либо остроугольными, либо прямоугольными, либо тупоугольными, и, если да, указать их вид.

11.53. Программа. Даны координаты трех вершин треугольника и координаты некоторой точки внутри него. Найти расстояние от данной точки до ближайшей стороны треугольника.

11.54. Программа. Найти наименьшее общее кратное четырех заданных натуральных чисел. (*Подсказка*: выразить эту величину через наибольший общий делитель.)

✓ **11.55.** Программа. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (таковы, например, числа 41 и 43). Определить все пары «близнецов» из отрезка $[2, n]$, где n – заданное целое число, большее 2.

11.56. Программа. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей другого, за исключением его самого (таковы, например, числа 220 и 284). Вывести все пары «дружественных» чисел, не превосходящих заданного натурального числа.

11.57. Программа. Даны вещественные коэффициенты многочленов $P(x)$ и $Q(x)$ n-й степени (n=15) и число a. Вычислить величину $P(a+Q(a)P(a+1))$.

11.58. Программа. По вещественному числу a>0 вычислить величину

$$\frac{\sqrt[3]{a}-\sqrt[6]{a^2+1}}{1+\sqrt[7]{3+a}}$$

Корни $y=\sqrt[k]{x}$ вычислять с точностью ε=0.0001 по следующей итера-

ционной формуле (возведение в степень вычислять через умножение):

$y_0=1$; $y_{n+1}=y_n+(x/y_n^{k-1}-y_n)/k$ $(n=0, 1, 2, \ldots)$,

приняв за ответ приближение y_{n+1}, для которого $|y_{n+1}-y_n|<\varepsilon$.

11.59. Программа. По вещественным числам $\varepsilon>0$ и t вычислить с точностью ε величину

$$\sqrt[4]{1-\frac{\cos^4 t}{4}}+\sqrt[5]{1+\frac{\operatorname{arctg} t}{2}}\cdot\sqrt[9]{\frac{1}{3+t^2}}$$

Для вычисления корней использовать следующий ряд Тейлора:

$$(1+x)^a=1+\frac{a}{1!}x+\frac{a(a-1)}{2!}x^2+\frac{a(a-1)(a-2)}{3!}x^3+\ \ldots\ \ (|x|\le 1, a>0).$$

11.60. Программа. Даны три целые матрицы размером $n\times m$ (n=9, m=4). Вывести на экран ту из них, где больше нулевых строк (если таких матриц несколько, вывести их все).

11.61. Программа. Даны натуральное число p и вещественные квадратные матрицы A, B и C n-го порядка (n=4). Получить $(ABC)^p$.

11.62. Программа. Даны вещественные матрицы A, B и C размером $n\times m$ (n=10, m=20). Вычислить величину

$$\frac{\|A\|+\|B\|+\|C\|}{\|A+B+C\|},$$

где $\|D\|=\max|D_{1,j}|+\max|D_{2,j}|+\ldots+\max|D_{10,j}|$.

11.63. Программа. Даны две целые квадратные матрицы n-го порядка (n=10). Определить, можно ли отражениями относительно главной и побочной диагоналей преобразовать одну из них в другую.

11.64. Программа. В точках 1, 2, ..., k, где k – заданное целое число от 2 до 70, вывести на экран (по отдельности) графики следующих функций:

$\varphi(n)$ – количество целых чисел от 1 до n–1, взаимно простых с числом n;

$\tau(n)$ – количество положительных делителей числа n;

$\pi(n)$ – количество простых чисел, не превосходящих n.

√ **11.65.** Программа. Дано n вещественных чисел (n=100). Упорядочить их по неубыванию *методом фон Неймана*: завести два массива A и B и записать исходные числа в A; упорядочить пары соседних чисел (A_1 и A_2, A_3 и A_4 и т.д.) и записать их в B; взять из B по две соседние упорядоченные пары и, слив их в упорядоченные четверки,

снова записать в A; затем каждые две соседние четверки из A слить в упорядоченные восьмерки и перенести в B; и т.д.

12. РЕКУРСИЯ

12.1*. Любое ли появление имени определяемой функции (процедуры) в ее описании означает рекурсию? Какая из следующих функций (обе – без параметров) является рекурсивной, а какая – нет?

```
function g: integer;
  var x: integer;
 begin read(x); if x<=0 then g:=0 else g:=x end;
function h: integer;
  var x: integer;
 begin read(x); if x<=0 then h:=0 else h:=x+h end;
```

Вычислить значение функции g, если для ввода заданы числа 1, 2 и 0. При тех же входных данных вычислить значение функции h.

12.2*. Вычислить $f(3)$ при следующем описании функции f:

```
function f (n: integer): integer;  {n≥0}
 begin if n=0 then f:=1 else f:=n*f(n–1) end;
```

12.3. На примере рекурсивной функции $f(n)$, вычисляющей факториал числа n (см. упр. 12.2) ответить на следующие вопросы.

а) Рекурсивность – это свойство самой функции или только ее описания? Можно ли функцию $f(n)$ описать нерекурсивно (циклически)? Любую ли рекурсивную функцию можно описать нерекурсивно? В чем достоинства и недостатки рекурсивных функций по сравнению с нерекурсивными?

б) Почему в рекурсивном описании функции должна быть хотя бы одна нерекурсивная ветвь, где значение функции дается явно, без повторного обращения к ней? Вычислить $f(2)$ при следующем описании функции f:

```
function f (n: integer): integer;
 begin f:=n*f(n–1) end;
```

в) Может ли в рекурсивном описании функции быть более одной нерекурсивной ветви? Определить, достаточно ли одной нерекурсивной ветви в следующем описании функции f:

```
function f (n: integer): integer;
 begin if n=0 then f:=1 else f:=n*(n–1)*f(n–2) end;
```

Вычислить $f(2)$ и $f(3)$ при этом описании.

г) Как узнать, сколько нерекурсивных ветвей должно быть в рекурсивном описании функции? Объяснить, какие нерекурсивные ветви лишние в следующем описании функции *f*:

```
function f (n: integer): integer;
 begin
  if n=0 then f:=1 else
   if n=1 then f:=1 else
    if n=2 then f:=2 else f:=n*f(n–1)
 end;
```

д) Рекурсивной называется ветвь, в которой имеется повторное обращение к определяемой функции. Каким должен быть параметр в этом повторном обращении – более простым или более сложным по сравнению с параметром, для которого дается определение функции? Вычислить *f*(3) при следующем описании функции *f*:

```
function f (n: integer): integer;   {n! = (n+1)! / (n+1)}
 begin if n=0 then f:=1 else f:=f(n+1) div (n+1) end;
```

е) Локализуются ли в повторном обращении к рекурсивной функции ее параметры-значения и вспомогательные объекты, описанные в ней, или они одинаковы во всех обращениях в этой функции? Если, к примеру, функция *f*(*n*) описана следующим образом:

```
function f (n: integer): integer;
  var k: integer;
 begin
  if n=0 then begin f:=1;  n:=8 end
         else begin k:=f(n–1);  f:=k*n end
 end;
```

то чему равно значение *n* (1 или 8) в операторе *f*:=*k***n* при вычислении *f*(1)?

12.4*. Рекурсивно описать функцию *f2*(*n*), вычисляющую двойной факториал *n*!! (*n*≥0), где:

$$n!! = \begin{cases} 1 & \text{при } n = 0 \\ 1 \cdot 3 \cdot \ldots \cdot (n-2) \cdot n & \text{при нечетном } n \\ 2 \cdot 4 \cdot \ldots \cdot (n-2) \cdot n & \text{при четном } n \end{cases}$$

12.5. Рекурсивно описать функцию *cos1*(*x*,*k*) от вещественного *x* и неотрицательного целого *k*, где:

cos1(*x*,*k*) = *cos*(*cos*(…*cos*(*x*)…)) (*k* косинусов).

12.6*. Описать рекурсивную функцию *pow*(*x*,*n*) от вещественного

1232

x (*x*≠0) и целого *n*, которая вычисляет величину x^n согласно формуле

$$x^n = \begin{cases} 1 & \text{при } n = 0 \\ 1/x^{|n|} & \text{при } n < 0 \\ x \cdot x^{n-1} & \text{при } n > 0 \end{cases}$$

12.7. Рекурсивно описать функцию *H*(*x*,*n*) от любого вещественного *x* и целого *n*≥0, где:

$$H(x,n) = \frac{x^n}{n!}$$

12.8*. Рекурсивно описать функцию *C*(*m*,*n*), где 0≤*m*≤*n*, для вычисления биномиального коэффициента C_n^m по следующей формуле:

$$C_n^0 = C_n^n = 1; \qquad C_n^m = C_{n-1}^m + C_{n-1}^{m-1} \quad \text{при } 0<m<n.$$

12.9. Рекурсивно описать функцию Аккермана *A*(*n*,*m*) от неотрицательных целых *n* и *m*, где:

$$A(n,m) = \begin{cases} m+1 & \text{при } n = 0 \\ A(n-1,1) & \text{при } n > 0,\ m = 0 \\ A(n-1, A(n,m-1)) & \text{при } n > 0,\ m > 0 \end{cases}$$

12.10*. Функция Маккарти определяется следующим образом:

```
function f (n: integer): integer;
 begin if n>100 then f:=n–10 else f:=f(f(n+11))  end;
```

Вычислить *f*(106), *f*(99) и *f*(85). Какие вообще значения принимает эта функция?

12.11. Если требуется описать какую-то функцию (или процедуру) рекурсивно, то возникают следующие вопросы: как «поймать» рекурсию, как ею воспользоваться, какие нерекурсивные ветви должны быть в описании?

На примере рекурсивного описания функции *sum*(*N*), вычисляющей сумму цифр неотрицательного целого числа *N*, ответить на следующие вопросы.

а) Повторное обращение к определяемой функции в ее описании означает решение подзадачи, которая аналогична исходной задаче (ее решение описывает функция) и которая, как правило, более простая, чем исходная задача. Поэтому, чтобы «поймать» рекурсию и применить ее, нужно свести исходную задачу к одной или нескольким более простым аналогичным подзадачам и для их решения

рекурсивно обратиться к определяемой функции, предполагая, что функция правильно решает эти подзадачи.

Если исходная задача – вычисление суммы цифр числа N, то является ли аналогичной и более простой подзадача вычисления суммы цифр числа, полученного из N отбрасыванием, например, последней цифры? Что означает величина $sum(N\ div\ 10)$?

б) Решив с помощью повторных обращений к определяемой функции более простые аналогичные подзадачи, нужно из ответов этих подзадач сконструировать ответ для исходной задачи.

Каким образом, зная сумму цифр числа $N\ div\ 10$, можно получить сумму уже всех цифр числа N? Является ли величина $sum(N\ div\ 10) + N\ mod\ 10$ ответом функции *sum* при параметре N?

в) Для выявления нерекурсивных ветвей нужно установить, при каких значениях параметра функции исходную задачу нельзя свести к более простым аналогичным подзадачам, и потому приходится давать явные ответы.

Например, при каких значениях параметра N нельзя свести вычисление суммы цифр числа к вычислению суммы всех цифр, кроме последней? Какие явные ответы должна выдавать функция *sum* при этих значениях параметра?

г)* Привести полностью рекурсивное описание функции $sum(N)$.

12.12. Пусть N – неотрицательное целое число. Рекурсивно описать функцию или процедуру от параметра N, которая:

а) подсчитывает количество (значащих) цифр числа N;

б) находит старшую (левую) цифру числа N;

в) находит наибольшую цифру числа N;

г) определяет, входит ли цифра 5 в запись числа N;

д) подсчитывает количество вхождений цифры 8 в запись числа N;

е)* выводит на экран цифры числа N в обратном порядке;

ж) используя только символьный вывод, выводит на экран число N;

з) подсчитывает сумму цифр, стоящих на нечетных местах в записи числа N.

12.13*. Программа. Во входном файле задана непустая последовательность положительных вещественных чисел, за которой следует отрицательное число. Описав подходящую рекурсивную функцию или процедуру, найти сумму этих положительных чисел.

12.14. Программа. Во входном файле задана непустая последовательность положительных целых чисел, за которой следует 0. Описав подходящую рекурсивную функцию или процедуру, найти наибольшее из этих чисел.

12.15. Программа. Во входном файле задан символьный текст, за которым следует точка. Описав подходящую рекурсивную функцию или процедуру, подсчитать количество цифр в данном тексте.

12.16. Программа. Во входном файле задан символьный текст, за которым следует точка. Описав подходящую рекурсивную функцию или процедуру, вывести этот текст в обратном порядке (без точки).

12.17. Программа. Во входном файле задана последовательность ненулевых целых чисел, за которой следует 0. Описав подходящую рекурсивную функцию или процедуру, вывести сначала все отрицательные числа этой последовательности, а затем – все положительные (в любом порядке).

12.18. Описать рекурсивную функцию *НОД*(a,b), вычисляющую наибольший общий делитель натуральных чисел a и b.

12.19. Описать рекурсивную функцию *root*(f,a,b,eps), которая методом деления отрезка пополам находит с точностью *eps* корень уравнения $f(x)=0$ на отрезке $[a,b]$. (Считать, что $eps>0$, $a<b$, $f(a)\cdot f(b)<0$ и $f(x)$ – непрерывная функция, имеющая на отрезке $[a, b]$ один корень.)

12.20*. const n=40;
type вектор = array[1..n] of real;
Описать функцию *min*(x) для определения минимального элемента вектора x, введя вспомогательную рекурсивную функцию *min1*(k), находящую минимум среди последних элементов вектора x, начиная с k-го.

12.21. type строка = packed array [1..100] of char;
Описать рекурсивную логическую функцию *симм*(s,i,j), которая проверяет, симметрична или нет часть строки s, начинающаяся i-м и кончающаяся j-м ее элементами.

12.22. type строка = packed array [1..100] of char;
Описать рекурсивную процедуру *обмен*(s,i,j), которая переставляет в обратном порядке символы строки s, начиная с i-го и кончая j-м символами.

12.23*. type имя = (Алла, ..., Юрий, нет);
Предполагая уже описанными функции *Отец*(x) и *Мать*(x), значениями которых являются имена соответственно отца и матери человека по имени x или идентификатор *нет*, если отсутствуют сведения о соответствующем родителе, описать логическую функцию *Пото-*

мок(*a*,*b*), проверяющую, является ли человек с именем *b* потомком (ребенком, внуком, правнуком и т.п.) другого человека с именем *a*.

(*Рекомендация*. Обобщить функцию *Потомок* следующим образом: при *a=нет* или *b=нет* функция должна выдавать значение *false*, а при *a=b* – значение *true*.)

12.24. Решить предыдущую задачу в предположении, что имеются функция *ЧислоДетей*(*x*), указывающая число детей человека с именем *x*, и функция *Ребенок*(*x*,*k*), указывающая имя *k*-го ребенка человека с именем *x* (*k* не должно превышать число детей человека *x*).

12.25*. Программа. Во входном файле записана формула, определяемая следующим образом:

<формула> ::= <цифра> | (<формула> <знак> <формула>)
<знак> ::= + | – | *
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Ввести эту формулу и вычислить ее значение. (Например:5→ 5, ((2–4)*6) → –12.)

12.26. Программа. Во входном файле записана формула, определяемая следующим образом:

<формула> ::= <цифра> | M(<формула>,<формула>) |
m(<формула>,<формула>)
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

где *M* означает максимум, а *m* – минимум. Ввести эту формулу и вычислить ее значение. (Например: *M*(4,*m*(5,8)) → 5.)

12.27. Программа. Во входном файле записано логическое выражение, определяемое следующим образом:

<логическое выражение> ::= true | false | <операция> (<операнды>)
<операция> ::= not | and | or
<операнды> ::= <операнд> | <операнд>, <операнды>
<операнд> ::= <логическое выражение>

(у операции *not* только один операнд, у *and* и *or* – не менее двух). Ввести это выражение и вычислить его значение. (Например: *and*(*or*(*false*,*not*(*false*)),*true*,*not*(*true*)) → *false*.)

12.28. Обычная запись бинарной (двухместной) операции *aΔb*, когда знак операции ставится между операндами, называется инфиксной формой записи. В постфиксной же форме знак операции ставится после операндов: *abΔ*. Возможные примеры:

инфиксная	*постфиксная*
x	x
x+y–z	xy+z–
x*(y+z)	xyz+*
x*y+z	xy*z+

(*Замечание*: в постфиксной записи скобки не ставятся.)

Пусть во входном файле записано в инфиксной форме выражение, определяемое следующим образом:

<выражение> ::= <терм> | (<выражение> <знак> <выражение>)
<знак> ::= + | – | *

где *терм* – любая буква. Написать программу, которая вводит это выражение и выводит его в постфиксной форме.

12.29. Рекурсия называется прямой, если при описании рекурсивной функции (процедуры) происходит обращение к этой же функции. Рекурсия называется косвенной, если при описании, скажем, функции *f* происходит обращение к функции (процедуре) *g*, которая непосредственно сама или через другие функции (процедуры) обращается к функции *f*.

При косвенной рекурсии возникает проблема с порядком описания функций *f* и *g*: если первой описать функцию *f*, тогда в ней будет обращение к еще не описанной функции *g*, а если первой описать функцию *g*, то в ней будет обращение к неописанной функции *f*. В любом из этих случаев нарушается общее правило языка Паскаль о том, что нельзя пользоваться именем до его описания. Чтобы решить эту проблему, для косвенной рекурсии сделано исключение из данного правила – это так называемые опережающие (предварительные) описания функций и процедур.

Ответить на следующие вопросы об опережающих описаниях.

а) Важно ли, для какой функции (*f* или *g*) приводить опережающее описание?

б) Верно ли, что в опережающем описании функции (скажем, *f*) полностью выписывается заголовок этой функции, а вместо ее блока указывается слово *forward*?

в) После опережающего описания функции *f* можно описывать функцию *g*. Каким должно быть это описание – полным или опережающим (частичным)? Должно ли оно сразу следовать за опережающим описанием функции *f*?

г) После описания функции *g* следует продолжить описание функции *f*, в котором уже полностью выписывается блок этой функции. Надо ли в этом продолжении выписывать полный заголовок функции *f*? Если нет, то что указывается вместо заголовка?

12.30*. Описать на языке Паскаль функции $f(n)$ и $g(m)$ от неотрицательных целых параметров при следующем их определении:

$$f(n) = \begin{cases} 1 & \text{при } n = 0 \\ n + g(n-1) & \text{при } n > 0 \end{cases}$$

$$g(m) = \begin{cases} 1 & \text{при } m = 0 \\ f(m-1) + g(m-1) & \text{при } m > 0 \end{cases}$$

12.31*. Программа. Во входном файле записана формула, определяемая следующим образом (точка – признак конца формулы):

<формула> ::= <сумма> .
<сумма> ::= <слагаемое>| <слагаемое> <знак ±> <сумма>
<знак ±> ::= + | –
<слагаемое> ::= <число> | (<сумма>)

где *число* – непустая последовательность цифр. Ввести и вычислить эту формулу. (Например: 15–(2+(39–28–6))+1. → 9.)

(*Рекомендация*. Описать две взаимно рекурсивные функции *слаг*(*c*) и *сумма* (без параметров): функция *слаг* вводит (из начала еще не считанной части входного файла) одно слагаемое и вычисляет его значение, присваивая параметру *c* символ, следующий за этим слагаемым; функция же *сумма* вводит формулу до закрывающей скобки или точки (включительно) и вычисляет ее значение.)

12.32. Программа. Во входном файле задан текст, за которым следует точка. Проверить, удовлетворяет ли его структура следующему определению:

<текст> ::= <элемент> | <элемент> <текст>
<элемент> ::= a | b | (<текст>) | [<текст>] | {<текст>}

(*Рекомендация*: описать две взаимно рекурсивные функции, первая из которых проверяет правильность одного элемента, а вторая – правильность текста.)

12.33. («Ханойские башни»). Имеются три колышка *A, B* и *C* и *n* дисков разного размера, перенумерованных от 1 до *n* в порядке возрастания их размеров. Сначала все диски надеты на колышек *A* так, как показано на рис. 6,*а*. Требуется перенести все диски с колышка *A* на колышек *C* (см. рис. 6,*в*), соблюдая при этом следующие условия: диски можно переносить только по одному, бóльший диск нельзя ставить на меньший.

Написать программу, которая выводит на экран последовательность действий (например, в виде «перенести диск с *q* на *r*», где *q* и *r* – это *A*, *B* или *C*), решающую указанную задачу для заданного натурального числа *n*.

(*Подсказка*: при правильном переносе *n* дисков с *A* на *C* обязательно встретится конфигурация, показанная на рис. 6,*б*.)

12.34*. Лабиринт из *n* комнат (*n*=20), некоторые пары которых соединены коридорами, можно представить в виде матрицы *L* следующего типа *лабиринт*:

```
const n=20;
type лабиринт = array[1..n,1..n] of 0..1;
```

где *L*[*i*,*j*]=*L*[*j*,*i*]=1 означает, что между *i*-й и *j*-й комнатами есть коридор, а *L*[*i*,*j*]=*L*[*j*,*i*]=0 – что такого коридора нет.

Описать логическую функцию *path*(*L*,*н*,*к*), определяющую, есть ли в лабиринте *L* путь из комнаты с номером *н* в комнату с номером *к*.

(*Замечания*. Учесть, что если есть коридор из комнаты *н* в комнату *r* и существует путь из комнаты *r* в комнату *к*, то существует и путь из *н* в *к*. Следует запоминать, какие комнаты уже посещались и снова не возвращаться в них, чтобы не было зацикливания.)

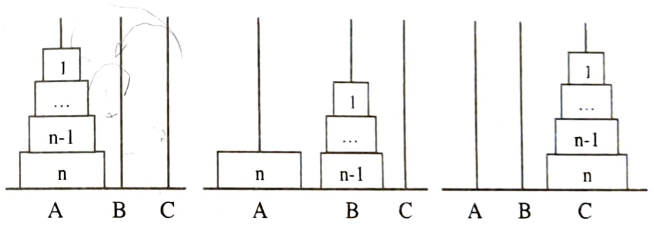


Рис. 6

12.35. Программа. Имеется *n* населенных пунктов, перенумерованных от 1 до *n* (*n*=10). Некоторые пары пунктов соединены дорогами. Определить, можно ли попасть по этим дорогам из 1-го пункта в *n*-й, и, если да, вывести на экран номера пунктов найденного пути.

Информация о дорогах задается в виде последовательности пар чисел *i* и *j* (*i*<*j*), указывающих, что *i*-й и *j*-й пункты соединены дорогой; признак конца этой последовательности – пара нулей.

12.36. Программа. Даны в порядке возрастания натуральные числа $a_1, a_2, \ldots, a_n$ (*n*=10), а также натуральное число *S*. Определить,

можно ли из этих a_i выделить такой набор чисел (без повторов), чтобы их сумма равнялась числу S, и, если можно, вывести числа найденного набора.

12.37. Программа. Даны натуральные числа m_1, m_2, …, m_n (m_1=1, m_i<m_{i+1}, n=10) и натуральное число S. Рассматривая m_i как достоинства монет и считая, что монет каждого достоинства имеется в неограниченном количестве, определить, каким минимальным числом этих монет можно получить сумму S, и вывести монеты такого набора.

12.38. Программа. Дано n различных натуральных чисел (n=5). Вывести на экран все перестановки этих чисел.

12.39. Задача о 8 ферзях: на шахматной доске расставить 8 ферзей так, чтобы они не «били» друг друга. Написать программу, которая находит:

а) одну из таких расстановок;

б) все 92 такие расстановки.

12.40. Программа. Найти расстановку 5 ферзей на шахматной доске, при которой они «бьют» все поля доски.

13. КОМБИНИРОВАННЫЕ ТИПЫ. ОПЕРАТОР присоединения

13.1. Ответить на следующие вопросы о комбинированных типах.

а) Как называются объекты комбинированных типов?

б) Верно ли, что запись – это единый объект, состоящий из частей, называемых полями? Фиксировано ли число полей в записи или оно может меняться в процессе выполнения программы?

в) Могут ли поля записи быть разных типов? Могут ли все поля записи быть одного и того же типа? Есть ли какие-нибудь ограничения на типы полей? Может ли, например, поле записи быть массивом или записью?

г) Верно ли, что каждое поле записи имеет индивидуальное имя? Могут ли имена полей записи совпадать с именами других объектов программы (например, переменных), с именем своей же записи, с именами полей в других записях?

д) Какие объекты входят в следующий комбинированный тип T:

```
type T = record a: char;  b: real;  c: boolean end ?
```

е) Эквивалентны ли следующие конструкторы комбинированного типа:

```
record x: 1..9;  y: 1..9;  z: real end  и  record x, y: 1..9;  z: real end ?
```


ж) В конструкторе комбинированного типа за описанием последнего поля (перед *end*) не надо ставить точку с запятой. А что будет, если ее все-таки поставить?

з) Какие начальные значения получают поля записи при ее описании?

и) Допускаются ли в языке Паскаль функции, значениями которых являются записи? Могут ли записи быть параметрами процедур и функций?

13.2*. Описать комбинированный тип для представления следующего понятия:

а) цена в рублях и копейках;

б)* время в часах, минутах и секундах;

в) дата (число, месяц, год);

г) адрес (город, улица, дом, квартира);

д) семинар (предмет, преподаватель, номер группы и расписание: день недели, часы, аудитория);

е) бланк требования на книгу (сведения о книге: шифр, автор, название; сведения о читателе: номер читательского билета, фамилия; дата заказа);

ж)* экзаменационная ведомость (предмет, номер группы, дата экзамена, 25 строчек с полями: фамилия студента, номер его зачетной книжки, оценка за экзамен).

13.3. Описать следующее понятие в виде массива или в виде записи, а если возможно, то в том и другом виде:

а) обозначение поля шахматной доски (a5, h8 и т.п.);

б) комплексное число;

в) точка в 50-мерном пространстве.

13.4. Ответить на следующие вопросы об операциях над записями.

а) Как в программе сослаться на всю запись?

б) Распространяется ли на комбинированные типы правило, что каждый конструктор типа определяет новый тип данных, отличный от всех других типов?

в)* Допускается ли присваивание записей как единых объектов? Обязательно ли при присваивании обе записи должны быть одного и того же типа?

Пусть, к примеру, имеются описания:

```
type T = record x: real;  y: char end;
var A, B: T;
    C, D: record y: char;  x: real end;
```

E: record x: real; y: char end;

Какие из следующих операторов присваивание правильные, а какие нет?

1) A:=B; 2) C:=D; 3) A:=C; 4) A:=E

г) Какие еще операции, кроме присваивания, применимы к записям как единым объектам? Можно ли, например, сравнивать их, вводить и выводить? Что делать, если нужно выполнить какую-то операцию над записями, а ее нет в языке Паскаль?

д) Пусть в записи с именем *R* первое поле называется *p*. Как сослаться на это поле: *R*[1], *R*.1 или *R.p*?

е) Если в записи *R* имеется поле *p*, то как определяется тип переменной (обозначения поля) *R.p*? Верно ли, что эта переменная может использоваться так же, как и любая другая переменная этого типа?

ж) В переменной с индексом в качестве индекса можно указать как явный индекс (например, *A*[1]), так и переменную (*A*[*i*]). А почему в обозначении поля (например, *R.p*) имя поля (*p*) должно указываться явно и не может задаваться переменной?

Поясните свой ответ на примере (неправильного) оператора *R.x*:=*R.x*+1 при следующих описаниях:

```
type имя = (p, q);
var x: имя; R: record p: integer; q: char end;
```

з) Имена полей записи могут совпадать с именами других объектов программы. Не ведет ли это к путанице?

Пусть, к примеру, имеются описания

```
type T = record p, q: char end;
var p: boolean; q: T;
```

Как записать обращения: к логической переменной *p*, к полю *p* записи *q*, во всей записи *q*, к полю *q* записи *q*?

13.5*. type цена = record руб: 0..maxint; коп: 0..99 end;
зарплата = record руб: 0..maxint; коп: 0..99 end;
var a, b: цена; c: зарплата; less: boolean;

Выписать фрагмент программы для решения следующей задачи.

а) Переменной *a* присвоить значение переменной *b*.

б) Переменной *a* присвоить значение переменной *c*.

в) Переменной *less* присвоить значение *true*, если зарплата *c* меньше цены *b*, и значение *false* иначе.

г) Переменной *a* присвоить в качестве значения цену *b*, увеличенную на 1 копейку.

д) Ввести два неотрицательных целых числа, второе из которых меньше 100, и, трактуя их соответственно как число рублей и

копеек, присвоить переменной *b*.

13.6. type масть = (пики, трефы, бубны, червы);
название = (шесть, семь, восемь, девять, десять,
валет, дама, король, туз);
карта = record м: масть; н: название end;

Описать логическую функцию *бьет*(*K1*,*K2*,*KM*), проверяющую, «бьет» ли карта *K1* карту *K2* при условии, что масть *KM* является козырной.

13.7*. type строка = packed array [1..15] of char;
вершина = record название: строка;
высота: 1000..9999 end;
список = array [1..30] of вершина;

Описать процедуру *СамаяВысокая*(*C*), которая выводит на экран название самой высокой вершины из списка *C*.

13.8*. type точка1 = array [(x,y)] of real;
точка2 = record x, y: real end;
var p1: точка1; p2: точка2; d: real;

Рассматривая *p1* и *p2* как точки на плоскости, присвоить переменной *d* расстояние между этими точками.

13.9. type КостьДомино = record лев, прав: 0..6 end;
ряд = array [1..28] of КостьДомино;

Описать логическую функцию *ПравильныйРяд*(*r*), которая проверяет, правильно ли выставлены кости домино в ряду *r* (равна ли правая цифра очередной кости левой цифре следующей кости).

13.10. type поле = record верт: (a, b, c, d, e, f, g, h); гориз: 1..8 end;

Описать логическую функцию *ХодФерзя*(*n1*,*n2*), проверяющую, может ли ферзь за один ход перейти с поля *n1* шахматной доски на поле *n2*.

13.11. type время = record час: 0..23; мин, сек: 0..59 end;

Описать:

а) логическую функцию *раньше*(*t1*,*t2*) для проверки, предшествует ли время *t1* времени *t2* (в рамках суток);

б) процедуру *интервал*(*d*,*t2*,*t1*), которая вычисляет время *d*, прошедшее от времени *t1* до времени *t2*: *d*=*t2*–*t1* (считать, что *t2*>*t1*).

13.12. type рац = record числ: integer; знам: 1..maxint end;
массив = array [1..20] of рац;

Описать:

а) логическую функцию *равно*(*a*,*b*), проверяющую на равенство два рациональных числа *a* и *b*;

б) процедуру *слож*(*c*,*a*,*b*), которая складывает рациональные

числа *a* и *b* и присваивает их сумму рациональному параметру *c*;

в) процедуру *max(x,m)*, присваивающую параметру *m* наибольшее из рациональных чисел массива *x*.

13.13. Ответить на следующие вопросы об операторе присоединения *with R do S*.

а) Что в этом операторе обозначают *R* и *S*?

б) В операторе присоединения за служебным словом *do* можно указать только один оператор. А что делать, если здесь надо указать несколько операторов?

в) Выполнение оператора присоединения сводится просто к выполнению оператора *S*. В чем же тогда выгода от оператора присоединения?

г) В операторе *S* ссылка на любое поле (скажем, *p*) записи *R* указывается без префикса «*R.*», т.е. вместо *R.p* указывается просто *p*. А можно ли в *S* писать *R.p*?

д)* Пусть имеются описания:

```
type T = record a: real;  b: array[boolean] of char end;
var x: array[1..10] of T;  i: 1..10;
```

Как записать оператор

```
with x[i] do begin a:=1.2;  b[true]:='*' end
```

без использования оператора присоединения?

е) Имена полей записи могут совпадать с именами других объектов программы, и это не ведет к путанице. Использование же оператора присоединения вносит путаницу.

Пусть, к примеру, имеются описания

```
var p: real;
    q, r: record p: integer; q, r: char end;
```

и имеется оператор присоединения

```
with r do begin p:=5;  q:=r end
```

Чему в этом операторе присваивается значение 5 – вещественной переменной *p* или полю *p* записи *r*? Что означает оператор *q:=r* – присваивание записей или присваивание полей записи *r*? Как в этом операторе присоединения сослаться на переменную *p* или запись *q*?

13.14*. var time: record час: 0..23; мин, сек: 0..59 end;

Присвоить переменной *time* значение, соответствующее времени 18 часов, 45 минут и 10 секунд, при следующем условии:

а) не используя оператор присоединения;

б) используя оператор присоединения.

13.15. type строка = packed array [1..8] of char;


```
адрес = record
          город, улица: строка;
          дом, квартира: 1..999
        end;
var Адр1, Адр2: адрес;
```

Используя оператор присоединения, присвоить переменной *Адр1* значение, соответствующее адресу «Москва, ул. Арбат, д. 1, кв. 5». Кроме того, переменной *Адр2* присвоить такое же значение, заменив в нем номер дома на 17.

13.16*. type время = record час: 0..23; мин, сек: 0..59 end;

Описать процедуру *следсек(t)*, увеличивающую значение времени *t* на 1 секунду с учетом смены суток.

13.17. type рац = record числ: integer; знам: 1..maxint end;

Описать процедуру *сокр(r)*, приводящую рациональное число *r* к несократимому виду;

13.18. type имя = (Аня, Валя, Женя, Петя, Саша, Таня, Шура, Юра);
данные = record пол: (муж, жен); рост: 140..200 end;
группа = array [имя] of данные;

Описать:

а) функцию *средрост(ГР)*, определяющую средний рост женщин из группы *ГР*;

б) функцию *высокий(ГР)* для определения имени самого высокого мужчины из группы *ГР*;

в) логическую функцию *одинрост(ГР)*, проверяющую, есть ли в группе *ГР* хотя бы два человека одного роста.

13.19. type число = 1..31; месяц = 1..12; год = 1..2099;
дата = record ч: число; м: месяц; г: год end;
деньнедели = (пн, вт, ср, чт, пт, сб, вс);

Считая, что все даты даются по григорианскому календарю («новому стилю»), описать:

а) функцию *послчисло(d)*, вычисляющую количество дней в том месяце, которому принадлежит дата *d*;

б) логическую функцию *вернаядата(d)*, проверяющую правильность даты *d* (т.е. чтобы не было 31 июня и т.п.);

в) функцию *числодней(d)*, подсчитывающую, сколько дней прошло от 1 января 1-го года нашей эры до даты *d*;

г) функцию *ДН(d)* для определения дня недели, на который приходится дата *d* (учесть, что 1 января 1-го года нашей эры было понедельником).

13.20. Ответить на следующие вопросы об операторе присоединения в общем виде.

а) Верно ли, что оператор присоединения

with R_1, R_2, …, R_n do S

является сокращенной формой записи следующего оператора:

with R_1 do
 with R_2 do
  …
  with R_n do S ?

На поля каких записей можно ссылаться в операторе *S*, не указывая имена этих записей и точку?

б) Пусть имеется описание

var R, Q: record a: char; b: 0..9 end;

Поле *a* какой записи (*R* или *Q*) имеется в виду в следующем операторе присоединения:

with R, Q do read(a) ?

Как в этом операторе присоединения сослаться на поле *a* записи *R*? Имеет ли смысл указывать между *with* и *do* однотипные записи?

в)* Пусть имеются описания

var R: record p: char; Q: a, b: integer end;

С каких из следующих заголовков может начинаться оператор присоединения, а с каких нет и почему?

1) with R, p do; 2) with R, Q do; 3) with Q, R do

13.21.
```
var a, b, c, d: real;
    p: record a, b: real;  c: record d: real end end;
    q: record a: real;  c: record d: real end end;
```

Каждый из следующих операторов записать в виде эквивалентного составного оператора, не содержащего операторов присваивания:

а)* with p do begin a:=b; d:=2 end;
б)* with q do begin a:=b; d:=2 end;
в)* with p, c do begin a:=b; d:=2 end;
г) with p, q do begin a:=b; d:=2 end;
д) with p, q, c do begin a:=b; d:=2 end

13.22*.
```
type круг = record радиус: real;
                   центр: record  x, y: real  end
            end;
var K: круг;
```

Требуется переменной *K* присвоить значение, соответствующее кругу радиуса 2.5 с центром в точке (0, 1.8). В каких из следующих операторов присоединения правильно решается эта задача, а в каких нет и почему?

а) with K do begin радиус:=2.5; x:=0; y:=1.8 end;
б) with K do begin радиус:=2.5; центр.x:=0; центр.y:=1.8 end;
в) with K do begin радиус:=2.5; with центр do
                                  begin x:=0; y:=1.8 end end;
г) with K, центр do begin радиус:=2.5; x:=0; y:=1.8 end;
д) with центр, K do begin радиус:=2.5; x:=0; y:=1.8 end

13.23.
```
type декарт = record x, y: real end;
     поляр = record r, fi: real end;   { r≥0, –π<fi≤π }
```

Описать процедуру:

а)* *ПД*(*p*,*d*), преобразующую координаты точки на плоскости из полярных *p* в декартовы *d*;

б) *ДП*(*d*,*p*), выполняющую обратное преобразование.

13.24.
```
type строка = packed array [1..16] of char;
     дата = record число: 1..31;  месяц: 1..12;
                год: 1..maxint end;

     анкета = record
                фамилия: строка;
                пол: (муж, жен);
                деньрожд: дата
              end;
     группа = array [1..25] of анкета;
```

Описать:

а)* процедуру *печ*(*Гр*,*Бук*), которая выводит на экран все фамилии людей из группы *Гр*, начинающиеся с символа *Бук*, и даты рождения этих людей;

б) функцию *моложе*(*Гр*,*d*), подсчитывающую количество женщин из группы *Гр*, родившихся позже даты *d*;

в) процедуру *старший*(*Гр*,*Фам*), присваивающую строке *Фам* фамилию самого старшего мужчины из группы *Гр* (считать, что такой есть и он единственный);

13.25.
```
type строка = packed array [1..20] of char;
     житель = record
                фамилия, город: строка;
                адрес: record улица: строка;
                       дом,  квартира: 1..999 end
```

```
                end;
        список = array [1..15] of житель;
```

Описать процедуру *ИронияСудьбы*(*С*), которая выводит на экран фамилии двух (любых) жителей из списка *С*, живущих в разных городах по одинаковому адресу (если такие есть).

13.26.
```
type слово = packed array [1..9] of char;
     НомерТелефона = 1000000..9999999;
     знакомый=record  фамилия: слово;
                      номер: НомерТелефона end;
     страница = array [1..20] of знакомый;
     ЗаписнаяКнижка = array ['A'..'Z'] of страница;
```

Считая, что на каждой странице записной книжки указаны фамилии, начинающиеся с одной и той же буквы – индекса этой страницы, описать логическую функцию:

а) *номер*(*ЗП*,*Ф*,*НТ*), определяющую, есть ли в записной книжке *ЗП* сведения о знакомом с фамилией *Ф*, и, если есть, присваивающую параметру *НТ* номер его телефона;

б) *фамилия*(*ЗП*,*НТ*,*Ф*), определяющую, есть ли в записной книжке *ЗП* сведения о знакомом, имеющем телефон с номером *НТ*, и, если есть, присваивающую параметру *Ф* фамилию этого знакомого.

13.27.
```
const n=300;
type запись = record  ключ: integer;
                      тело: array [1. .99] of char  end;
     таблица = array [1..n] of запись;
```

Считая, что в таблице записи имеют различные ключи, описать:

а) процедуру *упор*(*Т*), упорядочивающую записи таблицы *Т* по возрастанию их ключей;

б) логическую функцию *поиск*(*Т*,*К*,*И*), определяющую, есть ли в таблице *Т* (все записи которой уже упорядочены по возрастанию их ключей) запись с ключом *К*, и, если есть, присваивающую ее индекс параметру *И*.

13.28. Какие результаты будут выданы следующей программой?
```
program Prog (output);
  type TR = record a, b: integer end;
       TQ = record b, c: integer end;
  var R: TR;  Q: TQ;
  procedure P(X: TR; var Y: TQ);
    begin with X, Y do  begin a:=2;  b:=4;  c:=6  end end;
```



```
begin
  R.a:=1;  R.b:=3;  Q.b:=5;  Q.c:=7;
  P(R,Q);  writeln(R.a, R.b:3, Q.b:3, Q.c:3);
end.
```

13.29. Найти и объяснить ошибки в следующей программе:
```
program errors (input, output);
  type поле = (a, b);
       запись = record a: integer;  b: char end;
  var x, y: запись; c: char;
  function f (var z: запись): запись;
    var p: поле;
    begin for p:=a to b do f.p:=succ(z.p) end;
begin
  read(c);
  with x do begin a:=ord(c);  b:=c end;
  y:=x; if x=y then y:=f(x);
  with y do writeln(a,x)
end.
```

13.30.
```
type complex = record re, im: real end;
     coeff = record a, b, c: complex end;   {a≠0}
```

Описать процедуру *value*(*p*,*x*,*y*), которая вычисляет *y* – значение квадратного трехчлена ax^2+bx+c с комплексными коэффициентами из *p* в комплексной точке *x*.

13.31. Программа. Найти оба корня квадратного трехчлена с заданными комплексными коэффициентами (каждый коэффициент задается парой вещественных чисел).

13.32. Программа. Даны комплексное число *z* (пара вещественных чисел) и вещественное число ε>0. Вычислить с точностью ε значение следующей комплексной функции:

а) $e^z = 1+\frac{z}{1!}+\frac{z^2}{2!}+\ \ldots\ +\frac{z^n}{n!}+\ \ldots;$

б) $sh\,z = z+\frac{z^3}{3!}+\frac{z^5}{5!}+\ \ldots\ +\frac{z^{2n+1}}{(2n+1)!}+\ \ldots;$

в) $ch\,z = 1+\frac{z^2}{2!}+\frac{z^4}{4!}+\ \ldots\ +\frac{z^{2n}}{(2n)!}+\ \ldots;$

г) $\sin z = z - \frac{z^3}{3!} + \frac{z^5}{5!} - \ \ldots \ + (-1)^n \frac{z^{2n+1}}{(2n+1)!} + \ \ldots;$

д) $\cos z = 1 - \frac{z^2}{2!} + \frac{z^4}{4!} - \ \ldots \ + (-1)^n \frac{z^{2n}}{(2n)!} + \ \ldots;$

е) $\ln(1+z) = z - \frac{z^2}{2} + \frac{z^3}{3} - \ \ldots \ + (-1)^{n-1} \frac{z^n}{n} + \ \ldots \ (|z| < 1);$

ж) $arctg\, z = z - \frac{z^3}{3} + \frac{z^5}{5} - \ \ldots \ + (-1)^n \frac{z^{2n+1}}{2n+1} + \ \ldots \ (|z| < 1)$

13.33. Во входном файле содержится информация об итогах зимней сессии на 1 курсе. Сведения о каждом студенте-первокурснике (всего их 400) заданы в виде следующего текста:

<фамилия>, <номер группы>, <оценка1>, <оценка2>, <оценка3>

причем в фамилии – не более 12 букв, номер группы – целое от 101 до 116, каждая оценка – это 2, 3, 4 или 5, причем первая оценка – за экзамен по матанализу, вторая – по алгебре, третья – по программированию. Сведения о студентах отделены друг от друга точкой с запятой.

Написать программу, которая вводит эту информацию и выводит следующие данные:

а) фамилии студентов, имеющих задолженность хотя бы по одному предмету;

б) «качество» успеваемости, т.е. процент студентов, сдавших все экзамены на 5 и 4;

в) название предмета, который был сдан лучше всего;

г) номера групп в порядке убывания средней успеваемости их студентов.

13.34. Во входном файле записана следующая информация о каждом из 2000 студентов некоторого вуза:

<фамилия>,<имя>, <отчество>, <пол>, <возраст>, <курс>

причем в фамилии, имени и отчестве – не более 12 букв, пол указывается буквами М и Ж, возраст – целое от 16 до 35, курс – целое от 1 до 5. Сведения о студентах отделены друг от друга точкой с запятой.

Написать программу, которая вводит эту информацию и выводит следующие данные:

а) номер курса, на котором наибольший процент мужчин;

б) самые распространенные мужское и женское имена;

в) фамилии (в алфавитном порядке) и инициалы всех студентов, возраст и отчества которых являются одновременно самыми распространенными.

14. МНОЖЕСТВЕННЫЕ ТИПЫ

14.1. Ответить на следующие вопросы о множествах.

а) Можно ли в языке Паскаль использовать бесконечные множества?

б) Все элементы множества должны быть одного и того же типа, называемого базовым типом этого множества. Каким может быть этот базовый тип? Может ли это быть тип *integer*, тип *real* или составной тип? Допустимы ли множества из множеств?

в) В языке Паскаль множество задается с помощью конструктора множества $[\alpha_1, \ldots, \alpha_k]$, где α_i – выражение базового типа или диапазон $\beta..\gamma$ из двух выражений базового типа. Если α_i – выражение, то сколько и какие именно элементы оно «поставляет» в множество? А если это диапазон? Допустим ли диапазон $\beta..\gamma$ при $\beta>\gamma$ и, если да, что он означает?

г) Что означает конструктор множества [ ]?

д) Важен ли порядок перечисления элементов в конструкторе множества? Например, одинаковы или различны множества [1,2,3] и [3,2,1]?

е) Если в конструкторе множества какой-то элемент указан несколько раз, то сколько таких элементов будет в множестве? Например, одинаковы или различны множества [1,2,1,1] и [1,2]?

ж)* Пусть множественный тип *M* описан следующим образом:

type M = set of T;

где *T* – некоторый базовый тип. Какие множества входят в тип *M*? Если в базовом типе *T* имеется *n* различных значений, то сколько различных множеств будет в типе *M*: n^2, $n!$ или 2^n?

з) Верно ли, что пустое множество входит в любой множественный тип?

и) Какие начальные значения получают переменные множественного типа при своем описании?

к) Допускаются ли в языке Паскаль функции, значениями которых являются множества? Могут ли множества быть параметрами процедур и функций?

14.2*. type lets = set of (a, b, c);
Ответить на следующие вопросы.
а) Каков базовый тип у множеств типа *lets*?
б) Какие из множеств [*b*,*c*], [*c*,*b*], [*a*..*c*] и [*c*..*a*] входят в тип *lets*?
в) Явно перечислить все множества, входящие в тип *lets*.

14.3*. type ДеньНедели = (пн, вт, ср, чт, пт, сб, вс);
Описать множественный тип, включающий в себя все множества из:
а) названий любых дней недели;
б) названий рабочих дней недели.

14.4*. Какие из следующих типов описаны неправильно и почему?

```
type точки = set of real;
     байт = packed array [1..8] of 0..1;
     данные = set of байт;
     месяц = (янв, фев, мар, апр, май, июн, июл, авг,
              сен, окт, ноя, дек);
     M1 = set of месяц;
     M2 = set of июн..авг;
     M3 = set of дек..фев;
     M4 = set of (июн, июл, авг);
```

14.5*. Какие из следующих конструкторов множеств неправильные и почему?

а) [9, 6, 3, 0];	б) [2..3, 5, 7];	в) [1..15, 4..18];
г) [1, 1.41, 1.72, 2];	д) ['*', '*'];	е) ['*'..'*'];
ж) ['a'..'z'];	з) ['z'..'a'];	и) [odd(7), 0<2];
к) [2, sqrt(9)];	л) ['=', '>=', '>'];	м) [[ ], [5]]

14.6*. Какие множества определяют следующие конструкторы множеств при i=3 и j=5?
а) [i+3, j div 2, j..sqr(i)–3]; б) [2*i .. j]; в) [i..j, i..sqrt(j+4)]

14.7*. var s: set of char; c, d: char;
Переменной *s* присвоить:
а) пустое множество;
б) множество из малых гласных латинских букв (a, e, i, o, u);
в) множество из всех цифр;
г) множество символов, которые больше *c*, но меньше *d* (*c*<*d*).

14.8*. var A: set of 0..9; B: set of 8..15; C: set of '0'.. '9';
Обязательно ли при присваивании одного множества другому оба множества должны быть одного и того же типа? Какие из следующих присваиваний переменной *A* правильные, а какие нет и почему?

а) B:=[9,12]; A:=B; б) B:=[8,9]; A:=B; в) C:=[ ]; A:=C
Какому условию должны удовлетворять типы и значения множеств, чтобы одно из этих множеств можно было присвоить другому?

14.9. var A, B: set of 0..9; C: set of '0'.. '9'; D: set of 20..29; x: 0..9;
Ответить на следующие вопросы об отношениях между множествами.
а) Когда истинны отношения *A*=*B* и *A*<>*B*? Допустимо ли отношение *x*=[*x*]?
б) Что означает истинность отношений *A*<=*B* и *A*>=*B*? Всегда ли истинны отношения *A*<=*A* и [ ]<=*A*? Допустимо ли отношение *x*<=[*x*]?
в) Допустимы ли в языке Паскаль отношения *A*<*B* и *A*>*B*?
г)* Обязательно ли в отношении с множествами оба множества должны быть одного и того же типа? Какие из следующих отношений допустимы, а какие нет?
1) A<>C; 2) A=D; 3) A<=[15..20]; 4) A<=D
Какому условию должны удовлетворять базовые типы множеств, чтобы эти множества могли быть операндами одного отношения?
д) Что означает операция *x in A*? Каковы значения отношений *x in* [ ] и *x in* [*x*]?
е) Как правильно записать условие, что *x* не является элементом множества *A*: *x not in A*, *not x in A* или *not* (*x in A*)?
ж)* Обязательно ли в отношении *x in A* тип величины *x* должен совпадать с базовым типом множества *A*? Какие из следующих отношений правильные, а какие нет?
1) 5 in A; 2) 13 in A; 3) '0' in A; 4) '0' in []
Каково должно быть соотношение между типом величины *x* и базовым типом множества *A* в операции *x in A*?

14.10*. Вычислить значения отношений:

а) [2] <> [2, 2, 2];	б) ['a', 'b'] = ['b', 'a'];
в) [4, 5, 6] = [4..6];	г) ['c', b'] = ['c'..'b'];
д) [2, 3, 5, 7] <= [1..9];	е) [3, 6..8] <= [2..7, 9];
ж) [ ] <= ['0'..'9'];	з) 'q' in ['a'..'z'];
и) trunc(3.9) in [1, 3, 5];	к) odd(4) in [ ];
л) [2] < [1..3];	м) 66 = [66]

14.11. Эквивалентны ли следующие пары выражений?
а) p in [0, 5, 19] и (p=0) or (p=5) or (p=19);
б) p in [20..50] и (p>=20) and (p<=50)

14.12*. type строка = packed array [1..100] of char;
Описать функцию *счет*(*s*), подсчитывающую общее количество цифр и знаков '+', '–' и '*', входящих в строку *s*.

14.13*. type месяц = 1..12;
Описать функцию *ЧислоДней(m)*, определяющую количество дней в месяце *m* (невисокосного года).

14.14. Программа. Дано 100 целых чисел от 1 до 50. Определить, сколько из этих чисел являются числами Фибоначчи и у скольких из этих чисел первая значащая цифра равна 1 или 2.

14.15. В языке Паскаль нет стандартной операции, позволяющей узнать, сколько и какие именно элементы входят в некоторое множество. Как реализовать эту операцию с помощью других средств языка?

Пусть имеется описание

```
type Lat = set of 'a'..'z';
```

Описать:

а)* функцию *card(A)*, подсчитывающую количество элементов в множестве *A* типа *Lat* (например, *card*(['*a*', '*q*', '*s*'])=3):

б) процедуру *print(A)*, которая выводит на экран в алфавитном порядке все элементы множества *A* типа *Lat*.

14.16.
```
const n=10;
type номер = 1..n;
     матрица = array [номер, номер] of real;
     ном = set of номер;
```
Описать функцию *sum(A,s1,s2)*, вычисляющую сумму тех элементов матрицы *A*, номера строк и столбцов которых принадлежат соответственно непустым множествам *s1* и *s2* типа *ном*.

14.17*.
```
type ДеньНедели = (пн, вт, ср, чт, пт, сб, вс);
     РабочийДень = пн..пт;
var wd: ДеньНедели;  t: boolean;
```
Требуется переменной *t* присвоить значение *true*, если *wd* – рабочий день, и значение *false* иначе. Какими из следующих операторов правильно решается эта задача?

а) t:=wd in РабочийДень; б) t:=wd=РабочийДень;
в) t:=wd in [РабочийДень]; г) t:=wd in [пн..пт];
д) t:=[wd]<=[пн..пт]; е) t:=[wd]=[пн..пт]

14.18. Ответить на следующие вопросы о теоретико-множественных операциях.

а) Что является результатом операции пересечения множеств *A*B*? Всегда ли истинны отношения *A*B=B*A*, *A*A=A*, *A*[ ]=[ ]* и *A*B<=A*?

б) Что является результатом операции объединения множеств *A+B*? Всегда ли истинны отношения *A+B=B+A*, *A+A=A*, *A+[ ]=A* и *A<=A+B*?

в) Какие элементы входят в разность множеств *A–B*? Всегда ли истинны отношения *A–B=B–A*, *A–A=[ ]*, *A–[ ]=A*, *[ ]–A = [ ]*, *A–B<=A* и *A–B<=B*?

г)* Каково должно быть соотношение между базовыми типами двух множеств, чтобы к ним можно было применить операцию *, + или –?

Пусть, примеру, имеются описания:

```
var A: set of 0..9;  B: set of '0'.. '9';  C: set of 20..29;
```

Какие из следующих операций недопустимы и почему?

1) A*B; 2) A*[13..16]; 3) A*C;
4) A+[13..16]; 5) A+C; 6) [–1..10]–A

д) В каком порядке выполняются операции *, + и – в выражениях множественных типов?

14.19*. Вычислить выражения:

а) [1,3,5]+[2,4]; б) [1,3,5]*[2,4]; в) [1,3,5]–[2,4];
г) [1..6]+[3..8]; д) [1..6]*[3..8]; е) [1..6]–[3..8];
ж) [2, 4]+[1..5]; з) [2, 4]*[1..5]; и) [2, 4]–[1..5];
к) [ ]+[4]; л) [ ]*[4]; м) [ ]–[4]

14.20. Указать порядок выполнения операций и вычислить выражения:

а)* [2..13]*[3, 13..60] + [4..10] – [5..15]*[6];
б) [2..10] – [4, 6] – [2..12]*[8..15];
в) (['0'..'7']+['2'..'9']) * (['a']+['z'])

14.21. Упростить следующие множества (*A* и *B* - множества):

а)* *A*B–A*; б)* *A–(A–B)*;
в) *(A+B)–(A–B)–(B–A)*; г) *(A–B)+(B–A)+A*B*

14.22. Не используя дополнительных переменных, поменять местами значения переменных-множеств *A* и *B*.

14.23*. var x, y, z: set of 8..22;
Переменной *x* присвоить множество всех целых чисел от 8 до 22, переменной *y* – множество всех простых чисел из этого диапазона, а переменной *z* – множество всех составных чисел из этого же диапазона.

14.24. Программа. Дан текст из цифр и малых латинских букв, за которым следует точка. Определить, каких букв – гласных (a, e, i, o, u) или согласных – больше в этом тексте.

14.25*. var A, B: set of char; x: char;
Переменной *B* присвоить множество, полученное из *A*:

а) добавлением элемента *x*;

б) удалением элемента *x*.

Почему эти задачи нельзя решить с помощью операторов $B:=A+x$ и $B:=A-x$?

14.26. type натур = 1..maxint;

Описать:

а)* функцию *digits*(*n*), подсчитывающую количество различных (значащих) цифр в десятичной записи натурального числа *n*;

б) процедуру *print*(*n*), которая выводит на экран в возрастающем порядке все цифры, не входящие в десятичную запись натурального числа *n*.

14.27. Программа. Дан текст из малых латинских букв, за которым следует точка. Вывести:

а)* первое вхождение каждой буквы в текст, сохраняя их исходный порядок;

б) все буквы (по одному разу), входящие в текст не менее двух раз;

в) все буквы, входящие в текст по одному разу.

14.28. Программа. Дан текст, за которым следует точка. Вывести (по одному разу) все малые русские гласные буквы (а, е, и, о, у, ы, э, ю, я), входящие в этот текст.

14.29.
```
const n=20;
type T1 = array[1..n] of 0..999;
     T2 = array[1..n, 1..n] of 0..999;
```

Описать:

а) функцию *diff1*(*V*), подсчитывающую число различных элементов в массиве *V* типа *T1*;

б) функцию *diff2*(*M*), подсчитывающую число различных элементов в массиве *M* типа *T2*;

в) логическую функцию *same1*(*V*), определяющую, есть ли в массиве *V* типа *T1* хотя бы два одинаковых элемента;

г) логическую функцию *same2*(*M*), определяющую, есть ли в массиве *M* типа *T2* хотя бы два одинаковых элемента.

14.30. Найти и объяснить ошибки в следующем фрагменте программы:

```
type M = set of char;
function f (a, b: M;  x: char): M;
begin
  if a*b=0 then a:=[x] else
    if a<b then a:=b+x else
      if ord(x) in a–b then a:= a–[x..'<=']
```



```
    f:=a+b
  end;
```

14.31. Программа. Вывести в возрастающем порядке все целые числа из диапазона 1..10000, представимые в виде n^2+m^2, где $n, m \ge 0$.

14.32. Программа. Подсчитать количество различных целых чисел из диапазона 1..4900, которые представимы в виде n^2+2k^2, но не представимы в виде $7ij+j+3$ ($n, k, i, j \ge 0$).

14.33. Программа. Дано целое n от 2 до 1000. Используя метод «решето Эратосфена», вывести в убывающем порядке все простые числа из диапазона 2..*n*.

(Суть этого метода: выписываются все целые числа, большие 1; выбирается первое из них (это 2, простое число) и вычеркиваются все кратные ему числа, кроме него самого; затем берется следующее из невычеркнутых чисел (это 3, также простое число) и вычеркиваются все кратные ему, опять же кроме него самого; и так для каждого не вычеркнутого ранее числа. В конце концов останутся только простые числа, начиная с 2.)

14.34.
```
type продукт = (хлеб, масло, молоко, мясо, рыба, соль,
                сыр, колбаса, сахар, чай, кофе);
     ассортимент = set of продукт;
     магазины = array [1..20] of ассортимент;
```

Не используя операцию *in*, описать процедуру *Наличие*(*Маг*,*A*,*B*,*C*), которая по информации из массива *Маг* типа *магазины* (*Маг*[*i*] – это множество продуктов, имеющихся в *i*-м магазине) присваивает параметрам *A, B* и *C* типа *ассортимент* следующие значения:

A – множество продуктов, которые есть во всех магазинах;

B – множество продуктов, каждый из которых есть хотя бы в одном магазине;

C – множество продуктов, которых нет ни в одном магазине.

14.35.
```
type имя = (Вася, Володя, Ира, Лида, Марина, Миша,
            Наташа, Олег, Оля, Света, Юля);
     гости = set of имя;
     группа = array [имя] of гости;
```

Описать логическую функцию *Везде*(*ГР*), определяющую, есть в группе *ГР* хотя бы один человек, побывавший в гостях у всех остальных из группы (*ГР*[*x*] – множество людей, бывших в гостях у человека с именем *x*; $x \notin ГР[x]$).

14.36. type город = (a, b, c, d, e, f, g, h);

```
города = set of город;
рейсы = array [город] of города;
```

Описать процедуру *МожноПопасть*(*P*,*H*,*K*), которая по рейсам *P* (*P*[*x*] – множество городов, в которые можно за один рейс доехать автобусом из города *x*) определяет *K* – множество городов, в которые можно попасть автобусом (за один рейс или через другие города) из города *H*.

14.37. Описать логическую функцию *path*(*G*,*N*,*K*,*D*), которая определяет, есть ли в ориентированном графе *G* путь из вершины *N* в вершину *K*, и, если есть, присваивает параметру *D* длину (число дуг) кратчайшего пути из *N* в *K*.

Использовать следующее представление графа:

```
type вершина = (b1, b2, b3, b4, b5, b6, b7, b8);
соседи = set of вершина;
граф = array [вершина] of соседи;
```

(*G*[*x*] – множество вершин, в которые ведут дуги из вершины *x*.)

14.38. Программа. Дана непустая последовательность слов из малых русских букв; между соседними словами – запятая, за последним словом – точка. Вывести в алфавитном порядке:

а) все гласные буквы, которые входят в каждое слово;

б) все согласные буквы, которые не входят ни в одно слово;

в) все звонкие согласные буквы, которые входят хотя бы в одно слово;

г) все глухие согласные буквы, которые не входят хотя бы в одно слово;

д) все согласные буквы, которые входят только в одно слово;

е) все глухие согласные буквы, которые не входят только в одно слово;

ж) все звонкие согласные буквы, которые входят более чем в одно слово;

з) все гласные буквы, которые не входят более чем в одно слово;

и) все звонкие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово;

к) все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.

(*Примечание:* гласные буквы – *а*, *е*, *и*, *о*, *у*, *ы*, *э*, *ю*, *я* (*ё* обычно не входит в символьный тип); согласные – все остальные буквы, кроме *й*, *ь*, *ъ*; звонкие согласные – *б*, *в*, *г*, *д*, *ж*, *з*, *л*, *м*, *н*, *р*; глухие согласные – *к*, *п*, *с*, *т*, *ф*, *х*, *ц*, *ч*, *ш*, *щ*.)

14.39. Программа. Дан некоторый текст, за которым следует точка (в сам текст точка не входит). Определить, является ли этот текст правильной записью «формулы»:

```
<формула> ::= <терм> | (<формула> <знак> <формула>)
<знак> ::= + | – | *
<терм> ::= <имя> | <целое>
<имя> ::= <буква> | <имя> <буква> | <имя> <цифра>
<целое> ::= <цифра> | <целое> <цифра>
<буква> ::= а | б | в | г | д | е | ж
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

14.40. Программа. Вычислить определитель заданной квадратной матрицы *A* *n*-го порядка (*n*=15), используя формулу разложения по первой строке:

$$\det(A) = \sum_{k=1}^{n} (-1)^{k+1} a_{1k} \cdot \det(A_k),$$

где A_k – матрица, полученная из *A* удалением 1-й строки и *k*-го столбца. (*Рекомендация*: определить рекурсивную функцию от параметров *l* и *s*, которая по указанной формуле вычисляет определитель матрицы, полученной из *A* удалением первых *l* строк и всех столбцов, номера которых принадлежат множеству *s*.)

15. ФАЙЛОВЫЕ ТИПЫ

15.1. Ответить на следующие вопросы о файлах.

а) Все элементы файла должны быть одного и того же типа, называемого базовым типом файла. Какие ограничения накладываются на этот базовый тип? Например, могут ли элементы файла быть снова файлами или массивами, содержащими файлы?

б) Длина файла (число элементов в нем) может меняться. Может ли файл быть пустым, т.е. иметь длину 0? Есть ли ограничение на максимальную длину файла?

в) Указать, какие объекты входят в тип *T*, описанный следующим образом:

```
type T = file of 1..99;
```

г) Маркер файла указывает на текущую позицию файла. На сколько позиций (одну, две, ...) и в какую сторону (к началу или к концу файла) может сдвигаться этот маркер?

д) С каждым файлом связана буферная переменная. Если файл имеет имя *f*, то как обозначается эта переменная? Что хранится в ней? Каков ее тип? Может ли она использоваться так же, как и любая другая переменная этого типа? Например, можно ли ей присваивать?

е) Пусть в программе имеется описание файла *f*:

var f: file of char;

Нужно ли отдельно описывать буферную переменную этого файла? Какие значения получают при описании файл и его буферная переменная?

ж) Существует два режима работы с файлом – режим чтения и режим записи. Какой из этих режимов устанавливается для файла в самом начале выполнения программы? Можно ли одновременно и считывать из файла, и записывать в него? Можно ли менять режимы работы с файлом?

з) Можно ли присваивать один файл другому? Допустима ли операция присваивания для массивов или записей, в состав которых входят файлы? Почему файловые параметры процедур и функций не могут быть параметрами-значениями и обязательно должны описываться как параметры-переменные?

15.2*. Какие из следующих типов недопустимы и почему?

```
type A = array [1..15] of boolean;
     B = file of A;
     C = file of B;
     D = array[1..30] of B;
     E = record x: real;  y: file of char end;
```

15.3*.

```
type TF = file of 0..9;
var f, g: TF;
    r, s: record a: real;  b: TF end;
```

Какие из следующих операторов присваивания недопустимы и почему?

а) f:=g; б) f↑:=g↑; в) r:=s; г) f:=r.b; д) s.b↑:=g↑

15.4. Ответить на следующие вопросы об операциях чтения из файла *f* типа *file of T*.

а) Верно ли, что стандартная процедура *reset(f)* устанавливает для файла *f* режим чтения и теперь из файла можно считывать? На какую позицию файла указывает его маркер после выполнения этой процедуры? Каким становится значение буферной переменной *f*↑, если файл не пустой? А если он пустой?

б) Когда стандартная функция *eof(f)* выдает значение *true* – когда маркер файла *f* указывает на один из элементов файла *f* или когда маркер указывает на конец файла (на позицию за последним элементом)? Каково значение буферной переменной *f*↑, если *eof(f)=true*?

в) Как правильно переменной *x* присвоить текущий элемент файла: *x*:=*f* или x:=*f*↑? Какого типа должна быть переменная *x*?

г) Стандартная процедура *get(f)* сдвигает маркер файла. На сколько позиций (одну, две, …) и в какую сторону (к началу или к концу файла) сдвигает? Что становится значением буферной переменной *f*↑? Если перед выполнением *get(f)* маркер указывал на последний элемент файла, то на что будет указывать маркер после выполнения процедуры и каковы будут значения *f*↑ и *eof(f)*? Допустимо ли обращение к *get*, если маркер указывает на конец файла?

д) Верно ли, что выполнение стандартной процедуры *read(f,x)* эквивалентно выполнению операторов *x*:=*f*↑; *get(f)*? Как можно описать действие процедуры *read*, если не использовать понятия «маркер файла» и «буферная переменная»? На что указывает маркер файла после выполнения *read(f,x)* – на только что считанный элемент или на следующий элемент? Можно ли обращаться к процедуре *read* при *eof(f)=true*?

е) Верно ли, что в языке Паскаль считывать элементы из файла можно только последовательно друг за другом и всегда начиная с первого элемента файла?

15.5.

```
var f: file of integer;  x, y: integer;
```

Пусть файл *f* содержит три элемента – 1, 2 и 3. Определить, что будет выдано на экран в результате выполнения следующих операторов.

```
а)* reset(f);  get(f);  x:=f↑;  get(f);  y:=f↑; write(x, y:2);
б)* reset(f);  x:=f↑;  get(f);  y:=f↑; get(f);  write(x, y:2);
в)* reset(f);  read(f, x);  read(f, y);  write(x, y:2);
г)* reset(f);  read(f, x);  write(x, f↑:2);
д)  reset(f);  read(f, f↑) ;  write(f↑);
е)  reset(f);
      get(f);  read(f, x);
      if not eof(f) then read(f, y); if not eof(f) then read(f, y);
      write(x, y:2);
ж) reset(f);  y:=0;
      while not eof(f) do begin read(f, x);  y:=y+x end;
      write(x, y:2);
з)  reset(f);  y:=0;
      repeat read(f, x);  y:=y+x until eof(f);
      write(x, y:2)
```

15.6*. type слово = file of char;
Найти ошибки в следующем описании функции, которая должна определять количество элементов в произвольном файле типа *слово*:

```
function длина (w: слово): integer;
  var k: integer;  c: char;
begin
  reset(w);  k:=0;
  repeat read(w, c);  k:=k+1 until eof(w);
  длина:=k
end;
```

15.7*. type серия = file of real;
Описать функцию *отриц(s)*, подсчитывающую сумму отрицательных элементов в серии *s*.

15.8. type цена = record руб: 0..maxint; коп: 0..99 end;
прейскурант = file of цена;
Описать процедуру *min(П,Ц)*, присваивающую параметру *Ц* наименьшую цену из непустого прейскуранта *П*.

15.9. type слово = file of char;
Описать логическую функцию:
а)* *eq(w1,w2)*, проверяющую слова *w1* и *w2* на равенство;
б) *less(w1,w2)*, проверяющую, предшествует ли лексикографически слово *w1* слову *w2*.

15.10. type книга = file of char;
библиотека = array[1..1000] of книга;
Описать:
а)* функцию *СА(Б)*, определяющую, сколько книг в библиотеке *Б* начинается с буквы *'A'*;
б) логическую функцию *Есть(Б,К)*, проверяющую, есть ли в библиотеке *Б* книга *К*.

15.11. type запись = record ключ: integer; тело: file of integer end;
Описать функцию *входж(R)* для подсчета числа вхождений ключа записи *R* в тело этой записи.

15.12. type FC = file of char;
Описать:
а) логическую функцию *Баланс(t)*, проверяющую, сбалансирован ли файл *t* типа *FC* по круглым скобкам;
б) функцию *МаксУр(t)*, определяющую максимальный уровень вложенности круглых скобок в файле *t* типа *FC*, содержимое которого сбалансировано по круглым скобкам.

15.13. type ряд = file of integer;
Описать логическую функцию, проверяющую, образуют ли элементы непустого файла типа *ряд*:
а)* упорядоченную по возрастанию последовательность;
б) арифметическую прогрессию;
в) геометрическую прогрессию.

15.14. type FC = file of char;
Описать функцию *вхожд(t)*, подсчитывающую число вхождений пары *'AB'* в файл *t* типа *FC*.

15.15. type FR = file of real;
Описать функцию *предпосл(f)*, значением которой является предпоследний элемент файла *f*, имеющего тип *FR* и содержащего не менее двух элементов.

15.16. type FR = file of real;
Описать функцию *incr(f)* для подсчета числа элементов в наиболее длинной возрастающей подпоследовательности файла *f*.

15.17. Ответить на следующие вопросы об операциях записи в файл *f* типа *file of T*.

а) Верно ли, что стандартная процедура *rewrite(f)* устанавливает для файла *f* режим чтения и теперь в этот файл можно записывать? Сохраняет ли эта процедура содержимое файла *f* или уничтожает его? На какую позицию файла указывает его маркер после выполнения этой процедуры? Каково теперь значение буферной переменной *f*↑? Чему равно значение *eof(f)*?

б) Каково действие стандартной процедуры *put(f)*? Что она записывает в файл и в какую позицию? Как правильно записать новый элемент *x* в файл: *f*:=*x* или *f*↑:=*x*; *put(f)*? На что указывает маркер файла и каковы значения буферной переменной *f*↑ и функции *eof(f)* после выполнения процедуры *put(f)*?

в) Верно ли, что выполнение стандартной процедуры *write(f,x)* эквивалентно выполнению операторов *f*↑:=*x*; *put(f)*? Как можно описать действие процедуры *write*, если не использовать понятия «маркер файла» и «буферная переменная»? Каким здесь может быть параметр *x* (константой, переменной или выражением более общего вида) и какого типа?

г) Верно ли, что в режиме записи в файл *f* значение функции *eof (f)* всегда равно *true*? Имеет ли смысл пользоваться этой функци-

ей в данном режиме?

д) Верно ли, что в языке Паскаль запись в файл можно начать только с его первой позиции и что новые элементы можно записывать только в конец файла?

15.18*. var f: file of integer; i: integer;

Определить содержимое файла *f* после выполнения следующих операторов или указать ошибку в операторах.

а) rewrite(f); f↑:=1; put(f); put(f);
б) rewrite(f); f↑:=1; put(f); f↑:=2; put(f);
в) rewrite(f); write(f,1); write(f,2);
г) rewrite(f); write(f, f↑);
д) rewrite(f); f↑:=1; write(f,2);
е) rewrite(f);
if eof(f) then write(f, 1) else write(f, 2);
if eof(f) then write(f, 3) else write(f, 4);

15.19*. type строка = packed array [1..100] of char;
текст = file of char;

Описать процедуру *цифры*(*s*,*t*), записывающую в текст *t* все цифры из строки *s*.

15.20. type ряд = file of 1..maxint;

Описать процедуру *fib*(*f*,*n*), записывающую в ряд *f* все числа Фибоначчи (1, 1, 2, 3, 5, ...), не превосходящие целого положительного числа *n*.

15.21*. В языке Паскаль нет операции присваивания файлов. Как реализовать это действие с помощью имеющихся операций?

Пусть имеется следующее определение:

type FB = file of boolean;

Описать процедуру *присв*(*f*,*g*) от параметров типа *FB*, которая файлу *f* присваивает содержимое файла *g*.

15.22. type letters = file of 'a'..'z';

Описать процедуру *append*(*f*,*g*,*h*) от параметров типа *letters*, которая записывает в файл *f* сначала все элементы файла *g*, а затем – все элементы файла *h*.

15.23. type дата = record
месяц: (янв, фев, мар, апр, май, июн, июл,
авг, сен, окт, ноя, дек);
число: 1..31
end;
ФД = file of дата;

Описать процедуру *зап*(*d*,*s*,*w*) от параметров типа *ФД*, которая из файла *d* переписывает в файл *s* все летние даты, а в файл *w* – все зимние даты.

15.24. type книга = file of char;
библиотека = array[1..1000] of книга;

Описать процедуру *копир*(*Б1*,*Б2*), которая копирует в библиотеку *Б1* все книги из библиотеки *Б2*.

15.25. type FC = file of char;

Описать процедуру *перепись*(*f*,*g*) от параметров типа *FC*, которая переписывает в файл *f* содержимое файла *g*, оставляя из нескольких подряд идущих пробелов только один.

15.26*. При условии, что известен тип (*Ф*) файлов *f* и *g*, но не известен тип их элементов, описать процедуру *присв*(*f*,*g*), присваивающую файлу *f* содержимое файла *g*.

15.27. type FI = file of integer;

Пусть в каждом из файлов *f* и *g* элементы упорядочены по неубыванию. Требуется слить эти файлы в один файл *h*, также упорядоченный по неубыванию.

Решение этой задачи описать в виде процедуры *merge*(*f*,*g*,*h*) от параметров типа *FI*. (*Рекомендация*: использовать буферные переменные файлов *f* и *g*.)

15.28. Можно ли один и тот же файл несколько раз открывать на чтение? Что при этом происходит?

Пусть имеется следующее определение:

type FR = file of real;

Описать:

а)* функцию *less*(*f*), которая подсчитывает количество элементов непустого файла *f* типа *FR*, меньших среднего арифметического всех элементов этого файла;

б) функцию *nmax*(*f*), которая подсчитывает количество вхождений в непустой файл *f* типа *FR* его наибольшего элемента;

в) логическую функцию *mid*(*f*,*m*), которая определяет, имеет ли файл *f* типа *FR* нечетную длину, и, если имеет, присваивает своему параметру *m* средний элемент этого файла.

15.29. type человек =
record имя: packed array [1..9] of char; возраст: 1..99 end;
группа = file of человек;
имена = file of packed array [1..9] of char;

Описать процедуру *СамыеМолодые(ГР,ИМ)*, которая имена всех людей из непустой группы *ГР*, имеющих наименьший возраст, записывает в файл *ИМ* типа *имена*.

15.30. Программа. Дана непустая последовательность слов, содержащих от 1 до 8 букв; между соседними словами – запятая, за последним словом – точка. Вывести на экран все слова, отличные от последнего слова. (*Рекомендация*: использовать файл из 8-символьных строк.)

15.31. Программа. Дана непустая последовательность слов, содержащих от 1 до 8 букв; между соседними словами – запятая, за последним словом – точка. Вывести на экран все слова наименьшей длины.

15.32. С помощью стандартных процедур языка Паскаль нельзя менять количество и значения элементов, уже записанных в файл. А что делать, если по алгоритму необходимы такие изменения файла?

Пусть имеется следующее определение:

```
type текст = file of char;
```

Описать процедуру:

а)* *add1*(*t*,*c*), добавляющую символ *c* в начало текста *t*;

б) *addlast*(*t*,*c*), добавляющую символ *c* в конец текста *t*;

в) *double*(*t*), добавляющую за каждой цифрой текста *t* такую же цифру;

г) *replace*(*t*,*c*), заменяющую последний символ непустого текста *t* на символ *c*;

д) *next*(*t*), заменяющую в тексте *t* каждую цифру на следующую по величине цифру ('9' заменять на '0');

е) *delete*(*t*), удаляющую из текста *t* все символы '+' и '–';

ж) *del*(*t*), удаляющую из текста *t* предпоследний элемент, если такой есть;

з) *firsts*(*t*), оставляющую в тексте *t* только первые вхождения каждого символа.

15.33. Описать тип (любой из возможных) переменной *x* так, чтобы выражение *x*↑[5].*y*+[5] было правильным.

15.34. type T = file of char;

Рекурсивно описать целочисленную функцию *value*(*f*), которая вычисляет значение формулы, записанной (без ошибок) в файле *f* типа *T* и определяемой следующим образом:

```
<формула> ::= <цифра> | (<формула> <знак> <формула>)
<знак> ::= + | – | *
```



```
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

(*Предупреждение*: процедура *reset*(*f*) должна выполняться только один раз.)

15.35. type T = file of char;

Рекурсивно описать логическую функцию *same*(*f*), которая за один просмотр файла *f* типа *T* проверяет, есть ли в этом файле одинаковые элементы. (*Рекомендация*: использовать множество.)

15.36. Ответить на следующие вопросы о текстовых файлах (файлах типа *text*).

а) Является ли тип *text* стандартным? Если в программе имеется описание

```
var t: text;
```

то надо ли предварительно описывать тип *text*?

б) Каков базовый тип текстовых файлов? Каков тип буферной переменной *t*↑ текстового файла *t*? Какого типа должен быть параметр *c* в процедурах *read*(*t*,*c*) и *write*(*t*,*c*), если *t* – текстовый файл?

в) Текстовый файл делится на строки, каждая из которых – это последовательность символов, за которой следует элемент «конец строки»; число символов в строке (без учета «конца строки») называется длиной строки. Обязательно ли все строки файла должны иметь одинаковую длину? Допускаются ли пустые строки (длины 0)? Есть ли ограничение на максимальную длину строк? Кто определяет, на какие именно строки должен делиться текстовый файл?

г) Относится ли элемент «конец строки» к типу *char*? Фиксирует ли стандарт языка Паскаль, как именно реализуются «концы строк»? Почему не эквивалентны типы *text* и *file of char*? Чему равно значение буферной переменной текстового файла, если текущим элементом является «конец строки» (не относящийся к типу *char*)?

д) Для записи в текстовый файл *t* элемента «конец строки» надо обратиться к стандартной процедуре *writeln*(*t*). Почему такую запись нельзя произвести с помощью процедуры *write*? Обязательно ли записывать «конец строки» за последней строкой текстового файла?

е) Если по процедуре *reset*(*t*) открывается на чтение непустой текстовый файл *t* и если в конце этого файла нет элемента «конец строки», то данная процедура добавляет этот элемент в конец файла. Зачем это делается, в чем выгода от этого? Добавляет ли процедура *reset* «конец строки» в пустой текстовый файл?

ж) Каково значение стандартной функции *eoln*(*t*), если текущим элементом текстового файла *t* является «конец строки»? А если

текущим является какой-то иной элемент файла? Каково значение буферной переменной $t\uparrow$ в каждом из этих двух случаев? Допустимо ли обращение к *eoln*, если маркер файла указывает на конец файла, т.е. если *eof*(*t*)=*true*?

з) Верно ли, что стандартная процедура *readln*(*t*) пропускает все элементы текущей строки текстового файла *t* до ближайшего элемента «конец строки» и устанавливает маркер файла на позицию вслед за этим элементом? Если за текущей строкой есть еще одна строка, то на что будет указывать маркер файла в случае, когда эта следующая строка не пустая, и в случае, когда она пустая? На что будет указывать маркер, если следующей строки нет? Что будет, если перед *readln*(*t*) было *eoln*(*t*)=*true*? Допустимо ли обращение *readln*(*t*), если маркер файла указывает на конец файла, т.е. если *eof*(*t*)=*true*?

и) Какие еще файлы, кроме текстовых, могут делиться на строки? К каким еще файлам можно применять стандартные процедуры *writeln* и *readln* и функцию *eoln*?

15.37*. Описать процедуру *triangle*(*t*), формирующую текстовый файл *t* из 9 строк, в первой из которых – один символ '1', во второй – два символа '2', ..., в девятой – девять символов '9'.

15.38. type M = array[1..400] of char;
Описать процедуру *line40*(*x*,*t*), которая переписывает символы из массива *x* типа *M* в текстовый файл *t*, формируя в нем строки по 40 символов.

15.39*. var t: text; k: integer; c: char;
Требуется присвоить переменной *k* значение 1, если непустой файл *t* начинается с пустой строки.

Какие из следующих фрагментов правильно решают эту задачу, а какие нет и почему?

а) reset(t); if eoln(t) then k:=1;
б) reset(t); read(t,c); if eoln(t) then k:=1;
в) reset(t); read(t,c); if c=' ' then k:=1

15.40*. var t: text; k: integer;
Если в следующем фрагменте программы есть ошибка, то указать ее, а иначе определить, каким станет значение переменной *k*?

```
rewrite(t); write(t,'a'); write(t, 'b');
 {в файл t не записан «конец строки»}
reset(t); k:=0;
while not eoln(t) do begin k:=k+1; get(t) end
```


15.41*. var t: text; c: char;
Пусть маркер файла *t* указывает на элемент «конец строки». Требуется пропустить этот элемент файла.

Какие из следующих операторов правильно решают эту задачи?

а) get(t); б) read(t,c); в) readln(t)

15.42. Описать функцию, которая:

а)* подсчитывает количество пустых строк в текстовом файле;
б) находит максимальную длину строк непустого текстового файла;
в) проверяет на равенство два текстовых файла.

15.43. Описать процедуру *printlines*(*t*), которая построчно выводит на экран содержимое текстового файла *t*.

15.44. Пусть текстовый файл *t* разбит на непустые строки. Описать функцию *count*(*t*) для подсчета числа строк, которые:

а)* начинаются с буквы *'d'*;
б) оканчиваются буквой *'z'*;
в) начинаются и оканчиваются одним и тем же символом;
г) состоят из одинаковых символов.

15.45. Описать функцию, определяющую, со скольких пустых строк начинается текстовый файл.

15.46*. Описать процедуру *присв*(*t1*,*t2*), переписывающую в текстовый файл t1 содержимое текстового файла *t2* (с сохранением деления на строки).

15.47. Описать процедуру *присв*(*t1*,*t2*), переписывающую в текстовый файл *t1* содержимое текстового файла *t2*, но без пустых строк.

15.48. Считая, что непустой текстовый файл *f* разбит на строки, длина каждой из которых не превосходит 80, описать процедуру *преобр*(*f*,*f80*), которая, дополняя пробелами концы коротких строк файла *f*, формирует текстовый файл *f80*, все строки в котором имеют длину 80.

15.49. Описать процедуру, которая в текстовом файле заменяет:

а) каждый символ табуляции (он имеет код 9) на 8 пробелов;
б) каждую группу из 8 соседних пробелов на один символ табуляции.

15.50. Описать процедуру, которая:

а) в текстовом файле заменяет *k* подряд идущих одинаковых символов ($1 \le k \le 255$) на два символа – символ с кодом *k* и этот повторяющийся символ;

б) выполняет обратное преобразование.

15.51. Ответить на следующие вопросы.

а) Верно ли, что файлы ввода *input* и вывода *output* относятся к текстовым файлам? Означает ли это, что в языке Паскаль ввод и вывод рассматриваются как частный случай чтения из текстовых файлов и записи в них?

б) Почему в программах не надо указывать описание

var input, output: text;

и операторы

reset(input); rewrite(output) ?

в) Можно ли в Паскаль-программе явно указывать обращения *reset*(*input*), *rewrite*(*input*), *reset*(*output*) и *rewrite*(*output*)?

г) Верно ли, что обращения *read*(*x*) и *readln* являются сокращениями обращений *read*(*input*,*x*) и *readln*(*input*), а *write*(*x*) и *writeln* – сокращениями для *write*(*output*,*x*) и *writeln*(*output*)?

д) В общем случае процедура *read*(*f*,*x*) позволяет считать из файла *f* только один элемент, причем переменная *x* должна быть базового типа этого файла. Для текстового файла это означает, что переменная *x* должна быть типа *char* и что за один раз из файла можно считать только один символ. В то же время при вводе, т.е. при обращении *read*(*input*,*x*), можно считать за один раз не только символ, но и число, а переменная *x* может быть не только типа *char*, но и типа *integer* или *real*. Распространяется ли такая возможность на чтение из любого текстового файла?

е) При вводе можно указывать любое число параметров в процедурах *read* и *readln*. Распространяется ли такая возможность на чтение из любого текстового файла? Если *t* – произвольный текстовый файл, то верно ли, что

1) read(t,x_1, …,x_n) эквивалентно read(t,x_1); …; read(t,x_n);

2) readln(t,x_1, …,x_n) эквивалентно read(t,x_1, …,x_n); readln(t) ?

ж) В общем случае процедура *write*(*f*,*x*) позволяет записывать в файл *f* только один элемент, причем параметр *x* должен быть базового типа этого файла. Для текстового файла это означает, что параметр *x* должен быть типа *char* и что за один раз в файл можно записать только один символ. В то же время при выводе, т.е. при обращении *write*(*output*,*x*), можно вывести за один раз не только символ, но и число, логическое значение или строку, а параметр *x* может быть не только типа *char*, но и типа *integer*, *real*, *boolean* или строкового. Распространяется ли такая возможность на запись в любой текстовый файл?

з) При выводе *write*(*x*) параметр *x* может быть задан в одной из следующих трех форм: *e*, *e*:*n* и *e*:*n*:*m*. Допускаются ли такие же формы параметра при записи в любой текстовый файл?

Например, сколько и какие символы окажутся в текстовом файле *t* после выполнения операторов

rewrite(t); write(t, 12:4); write(3.14159:5)?

и) Для любого ли текстового файла *t* верно, что

1) write(t,x_1, …,x_n) эквивалентно write(t,x_1); …; write (t,x_n);

2) writeln(t,x_1, …,x_n) эквивалентно write(t,x_1,…,x_n); writeln(t) ?

15.52*. type слово = packed array [1..20] of char;

список = array [1..100] of слово;

Описать процедуру *zan*(*l*,*t*), записывающую слова списка *l* как строки в текстовый файл *t*.

15.53*. В каждой строке непустого текстового файла *t* записано одно вещественное число. Описать функцию *max*(*t*) для нахождения наибольшего из этих чисел.

15.54. В каждой строке текстового файла *t1* записаны через пробел два целых числа. Описать процедуру *positive*(*t1*,*t2*), которая переписывает в текстовый файл *t2* все положительные числа из *t1* (через пробел, в одну строку).

15.55. Описать процедуру *lines*(*t1*,*t2*), которая построчно копирует содержимое непустого текстового файла *t1* в текстовый файл *t2*, вставляя в начало каждой строки ее порядковый номер (он должен занимать 4 позиции) и пробел.

15.56. Ответить на следующие вопросы о внешних файлах.

а) Чем отличаются внешние файлы Паскаль-программы от других (внутренних) файлов?

б) Какие еще файлы, кроме файлов *input* и *output*, могут быть внешними?

в) Верно ли, что имена всех внешних файлов должны быть перечислены в заголовке программы? Как по тексту программы определить, какие файлы в ней внешние, а какие – внутренние? Возможна ли Паскаль-программа, в заголовке которой не указаны файлы *input* и *output*?

г) Отличается ли чем-нибудь работа с внешними файлами от работы с внутренними файлами? Например, надо ли описывать внешние файлы (кроме *input* и *output*)? Надо ли применять процедуры *reset* и *rewrite* к внешним файлам? С помощью каких процедур

осуществляется чтение из внешних файлов и запись в них?

15.57*. Имеется внешний текстовый файл *BOOK*. Написать программу, которая, игнорируя исходное деление этого файла на строки, переформатирует его, разбивая на строки так, чтобы каждая строка либо оканчивалась точкой, либо содержала ровно 60 символов, если среди них нет точки.

15.58. Программа. Пусть *T1* и *T2* – внешние текстовые файлы, причем в каждой строке файла *T1* записаны через пробелы целые числа. Переписать построчно содержимое *T1* в *T2* так, чтобы в каждой строке сначала шли ее положительные числа, а затем – все остальные.

15.59. Программа. Поменять местами содержимое внешних текстовых файлов *T1* и *T2*.

15.60. Программа. Во внешнем текстовом файле *PROG* записана (без ошибок) программа на языке Паскаль. Переписать ее в другой внешний текстовый файл *PROG1*, заменив на пробел каждый комментарий в ней, т.е. каждую последовательность, начинающуюся с символа '{' и заканчивающуюся ближайшим символом '}'. (Учесть, что в строках фигурные скобки являются обычными символами.)

15.61. Программа. Имеется внешний текстовый файл *T*. Вывести на экран первую из самых коротких его строк.

15.62. Программа. Подсчитать (с точностью до десятых) среднюю длину русских слов, входящих во внешний текстовый файл *DOCUM*. (Под словом понимать максимально длинную последовательность из подряд идущих больших и малых русских букв.)

15.63. Имеется внешний файл *COURSE* из элементов типа *stud*:

```
type str = packed array[1..10] of char;
     stud = record    {информация об одном студенте}
              fn: record fam, name: str end;  {фамилия, имя}
              sex: (M,W);          {пол: муж. (M), жен. (W)}
              marks: array[1..5] of 2..5;          {пять оценок}
            end;
```

Считая, что этот файл содержит от 1 до *N* элементов (*N*=30) и что фамилии и имена студентов записаны большими русскими буквами, составить программу для чтения данных из этого файла и записи во внешний текстовый файл *ANS* следующей информации (о каждом студенте – в отдельной строке):

а) о каждой студентке сообщить фамилию, имя и все ее оценки;

б) о каждом отличнике сообщить фамилию, имя и пол;

в) о каждом «хорошисте» (все оценки – 4 или 5 и есть хотя бы одна 4) сообщить фамилию, пол и все оценки;

г) о каждом студенте-юноше, имеющем хотя бы одну оценку 2, сообщить фамилию, имя и все оценки;

д) составить рейтинг – список всех студентов по убыванию их средних оценок, сообщив о каждом из них фамилию, имя и среднюю оценку (с одной цифрой в дробной части).

В вариантах *а*) – *г*) в ответе фамилии упорядочить по алфавиту.

15.64. Программа. Имеется внешний файл *A* из целых чисел. Используя еще три внешних файла *B*, *C* и *D* в качестве рабочих, упорядочить файл *A* по неубыванию следующим методом (называемым *внешней сортировкой сбалансированным слиянием).*

Назовем «отрезком» как можно более длинный упорядоченный по неубыванию участок файла (на рис. 7,*а* показан пример файла *A*, отрезки которого разделены вертикальными линиями). На начальном этапе сортировки определяются отрезки файла *A* и они попеременно переносятся в файлы *C* и *D* (рис. 7,*б*). На следующем этапе пары *i*-х отрезков файлов *C* и *D* (*i* = 1, 2, ...) сливаются в более длинные отрезки и попеременно переносятся в файлы *A* и *B* (рис. 7,*в*). Затем сливаются пары отрезков файлов *A* и *B* и переносятся в файлы *C* и *D* (рис. 7,*г*), и т.д. (Учесть, что в конце концов единая упорядоченная последовательность чисел должна оказаться в файле *A*.)

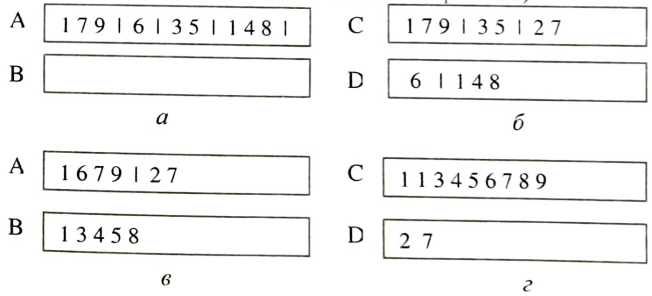


Рис. 7

16. ССЫЛОЧНЫЕ ТИПЫ. СПИСКИ

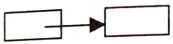
16.1. Ответить на следующие вопросы о ссылках.

а) Верно ли, что динамической называется переменная, которая может начать существовать в любой момент выполнения программы и может быть уничтожена в любое же время? Каких типов могут быть динамические переменные?

б) Верно ли, что число динамических переменных, которые могут появиться в программе, как правило, заранее не известно? Почему динамические переменные не описываются? Почему они не имеют имен? Как в языке Паскаль ссылаются на эти безымянные переменные? Верно ли, что ссылкой на динамическую переменную называют адрес того места в памяти компьютера, которое выделено под эту переменную? Имеют ли ссылки какое-нибудь внешнее представление в языке Паскаль?

в) Как называется переменная p, значением которой является ссылка на динамическую переменную? Что означает запись $p\uparrow$, которая называется переменной с указателем?

г) Ради наглядности ссылку обычно изображают стрелкой, ведущей от переменной, где хранится эта ссылка, к переменной, на которую она указывает:



Какая из этих двух переменных является ссылочной переменной, а какая – динамической?

д) В ссылочный тип данных входят ссылки на динамические переменные. Почему в языке Паскаль имеется не один, а много ссылочных типов? По какому признаку они различаются? Что означает T в конструкторе ссылочного типа $\uparrow T$? Если p – переменная ссылочного типа, то как узнать, допустим или нет оператор присваивания $p\uparrow$:=8.3?

е) В конструкторе ссылочного типа $\uparrow T$ в качестве T можно указывать только имя типа, но не конструктор типа. А как, например, описать ссылочный тип, состоящий из ссылок на динамические переменные ограниченного типа 0..9?

ж) Что такое пустая ссылка *nil*? Допустима ли запись $nil\uparrow$? Верно ли, что пустая ссылка входит в любой ссылочный тип?

з) Разрешены ли в языке Паскаль ссылки не на динамические

переменные, а на переменные, описанные в программе? Если, к примеру, имеются описания

```
var x: integer;  p: ↑integer;
```

то может ли значением переменной p быть ссылка на переменную x?

16.2*. Какие из следующих типов описаны неправильно и почему?

```
type  A = ↑char;
      B = array[1..10] of real;
      C = ↑B;
      D = ↑array[1..10] of real;
      E = ↑C;
      F = ↑↑B;
```

16.3. `var p, q: ↑T;  {T – некоторый тип}`

Ответить на следующие вопросы об операциях над ссылками.

а) Стандартная процедура *new*(p) создает новую динамическую переменную, отводя ей место в памяти компьютера. Какого типа эта переменная? Получает ли она какое-нибудь начальное значение? Должна ли ссылочная переменная p иметь какое-нибудь значение до выполнения процедуры *new*? Какое значение будет иметь переменная p после выполнения этой процедуры?

б) Что происходит при выполнении операторов

```
new(p);  p:=nil ?
```

Создает ли здесь процедура *new* переменную p или эта переменная уже должна существовать? Как после этих операторов сослаться на динамическую переменную, созданную процедурой *new*? Разумно ли с помощью этих операторов присваивать переменной p значение *nil*, как это иногда делают начинающие программисты?

в) Обязательно ли в операторе присваивания p:=q ссылочные переменные p и q должны быть одного и того же типа? Верно ли, что после такого присваивания обе переменные p и q будут ссылаться на одну и ту же динамическую переменную? Если переменной $q\uparrow$ присвоить какое-то значение, то каким будет значение переменной $p\uparrow$?

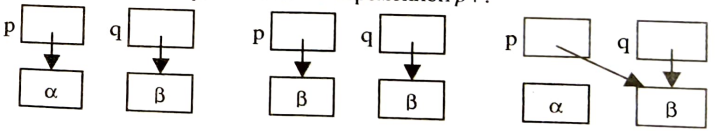


Рис. 8

г) Чем отличается присваивание p:=q от присваивания $p\uparrow$:=$q\uparrow$?

Если вначале переменные p и q имели значения, показанные на рис. 8 слева, то каким из этих двух операторов присваивания можно получить значения, показанные в середине, и каким – показанные справа?

д) Какие из операций сравнения (=, <>, <, <=, >, >=) применимы к ссылкам? Если $p\uparrow=q\uparrow$, то значит ли это, что $p=q$? А если $p=q$ и $p<>nil$, то значит ли это, что $p\uparrow=q\uparrow$?

е) Стандартная процедура *dispose*(p) уничтожает динамическую переменную, освобождая занимаемое ею место в памяти. Какую именно переменную она уничтожает – переменную p или переменную, на которую ссылается p? Каким должно быть значение переменной p до выполнения этой процедуры и каким оно станет после? Допустим ли, например, вывод *write*($p\uparrow$) после выполнения *dispose*(p)?

ж) Что такое «висячая ссылка»? Когда она появляется? Чем она опасна? Допустим ли вывод *write*($q\uparrow$) после выполнения операторов $q:=p$; *dispose*(p)?

з) Допускаются ли в языке Паскаль ссылочные функции, т.е. такие, значениями которых являются ссылки? Могут ли ссылки быть параметрами процедур и функций?

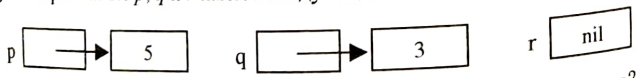
и) Можно ли вводить и выводить ссылки с помощью процедур *read* и *write*?

к) type R = record a: real; b: char end;
var p: ↑R;

Значением переменной p может быть ссылка на всю запись типа R. А можно ли в языке Паскаль каким-нибудь образом получить ссылку на отдельное поле этой записи? Что означает $p\uparrow.a$ – ссылку на поле a или само поле a?

16.4*. type ref = ↑integer;
var p, q, r: ref;

Пусть переменные p, q и r имеют следующие значения:



Что будет выдано на экран при выполнении следующих операторов?

```
r:=q; p↑:=q↑;
if p=q then p:=nil else if p↑=q↑ then q:=p;
if p=q then q↑:=4;
writeln(p↑, r↑:2)
```


16.5*. type D = record a: boolean; b, c: ↑real end;
var r: ↑D;

Пусть переменная r имеет значение, показанное на рис. 9.

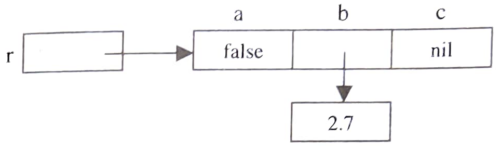


Рис. 9

а) Что обозначает каждая из следующих переменных: r, $r\uparrow$, $r\uparrow.b$ и $r\uparrow.b\uparrow$?

б) Нарисовать структуру значения переменной r после выполнения операторов:

```
if r↑.b<>nil then r↑.c:=r↑.b;
r↑.b↑:=r↑.c↑–1.4;  r↑.a:=r↑.b=r↑.c
```

16.6*. var p, q: ↑integer; r: ↑char;

Какие из следующих операторов неправильные и почему?

а) p:=q; б) q:=r; в) p:=nil; г) r:=nil;
д) q:=p↑; е) p↑:=nil; ж) r↑:=p↑; з) q↑:=ord(r↑);
и) if r<>nil then r↑:=nil↑; к) if q>nil then q↑:=p↑;
л) if q=p then write(q); м) if q<>p then read(r)

16.7. Имеется программа

```
program dynamic (output);
  var x: ↑boolean;  y: boolean;
begin
  {A} new(x);  {B} x↑:=true;  y:=not x↑;
  {C} dispose(x);  {D} writeln(y)
end.
```

Какие переменные (динамические и другие) существуют в каждой из точек A, B, C и D и каковы их значения в эти моменты?

16.8*. type A = ↑char; B = record f1: char; f2: A end;
var p: ↑B; q: A;

Нарисовать структуру значений переменных p и q после выполнения следующих операторов:

```
new(q);  q↑:='7';  new(p);  p↑.f1:=succ(q↑);  p↑.f2:= q
```

16.9*. Одно из общих правил языка Паскаль гласит, что использовать любое (нестандартное) имя можно только после того, как оно описано. Правда, из этого правила есть исключение – опережающие описания

функций и процедур. Верно ли, что имеется еще одно исключение: в конструкторе ссылочного типа ↑*T* можно указать имя *T*, которое еще не описано (но которое будет описано в том же разделе типов)?

Какие из следующих типов описаны неправильно и почему?

```
type  A1=array[1..4] of A2;   A2=↑A1;
      B1=↑B2;   B2='a'..'z';
      C1=↑C2;   C2=record x: real; y: C1 end;
      D1=record x: real; y: D2 end;   D2=↑D1;
      E=record x: real; y: ↑E end;
      F=↑F;
```

16.10.
```
type chain = ↑elem;
     elem = record data: integer;  link: chain end;
var p, q: chain;
```

Нарисовать структуру значения переменной *p* после выполнения следующих операторов:

а) new(p); p↑.data:=4; p↑.link:=nil;

б) new(p); p↑.data:=7; p↑.link:=p;

в) new(q); q↑.data:=2; q↑.link:=nil; new(p); p↑.data:=1; p↑.link:=q;

г)* new(p); p↑.data:=5; new(p↑.link); p↑.link↑:=p↑

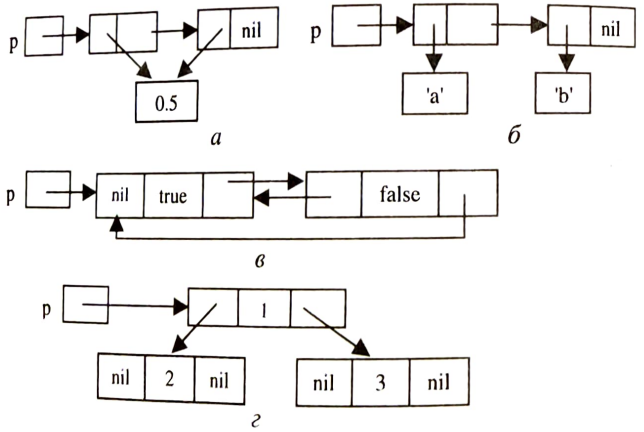


Рис. 10

16.11*. Описать переменную *p* (и, если надо, вспомогательные переменные) и выписать операторы, присваивающие ей значение,

указанное на рис. 10.

16.12.
```
type цепочка = ↑звено;
     звено = record элем: integer;  след: цепочка end;
var p: record нач, кон: цепочка end;
```

Выписать операторы, которые преобразуют значение переменной *p*, показанное на рис. 11,*а*, к значению, показанному: 1)* на рис. 11,*б*; 2) на рис. 11,*в*. (Ненужные звенья уничтожить.)

16.13. Определить, допустима ли в языке Паскаль следующая конструкция, и, если да, привести подходящее описание типа переменной *p*.

а) p↑[2];	б) p[2]↑;	в) p↑+[2];	г) [p↑+2];
д) p↑.p;	е) p.↑p;	ж) p.p↑;	з) p↑↑

16.14. Найти и объяснить ошибки в следующей программе.

```
program errors (input, output);
 var a, b: ↑integer;
begin
 if a=nil then read(a);  a↑:=5;
 b:=nil;  b↑:=2;
 new(b);  read(b↑);  writeln(b,b↑);
 new(a);  b:=a;  dispose(a);  b↑:=4
end.
```

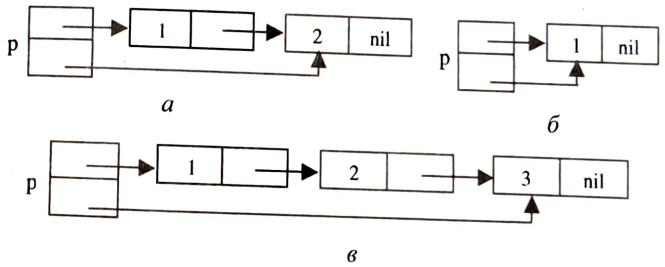


Рис. 11

16.15.
```
type ссылка = ↑real;
     вектор = array [1..100] of ссылка;
```

Считая, что все элементы вектора *x* отличны от *nil*, описать:

а) функцию *max*(*x*) для нахождения наибольшего из чисел, на которые ссылаются элементы вектора *x*;

б)* функцию *neg1*(*x*), значением которой является первый из элементов вектора *x*, ссылающихся на отрицательные числа, или *nil*,

если таких элементов нет;

в) логическую функцию *same*(*x*), которая проверяет, есть ли в векторе *x* хотя бы две одинаковые ссылки;

г) процедуру *unique*(*x*), которая в векторе *x* все элементы, ссылающиеся на равные числа, заменяет на первый из этих элементов.

16.16. Одно из возможных представлений «длинного» текста – это разделить его на участки (строки) равной длины и создать массив ссылок на эти строки:

```
const d=...;  {длина строки}
      n=...;  {максимальное число строк в тексте}
type строка = packed array [1..d] of char;
     ссылка = ↑строка;
     текст = array [1..n] of ссылка;
```

(Если в тексте менее *n* строк, то последние элементы массива равны *nil*; в начале массива ссылок *nil* не должно быть. Если в операции над текстом указан номер отсутствующей строки, т.е. элемент массива с этим номером равен *nil*, то такая операция не выполняется.)

Используя данное представление текста, описать:

а) функцию *числострок*(*T*) для подсчета числа строк в тексте *T*;

б)* логическую функцию *элем*(*T*,*i*,*j*,*c*), проверяющую, есть ли в тексте *T* строка с номером *i*, и, если есть, присваивающую *j*-й символ этой строки параметру *c*;

в) процедуру *перестановка*(*T*,*i*,*j*), меняющую местами *i*-ю и *j*-ю строки текста *T*;

г) процедуру *замена*(*T*,*i*,*j*), заменяющую *i*-ю строку текста *T* на копию *j*-й строки;

д) процедуру *добавить*(*T*,*i*,*j*), добавляющую после *i*-й строки текста *T* копию *j*-й строки;

е) процедуру *удалить*(*T*,*i*), удаляющую *i*-ю строку из текста *T*;

ж) логическую функцию *поиск*(*T*,*c*,*i*,*j*), определяющую, входит ли символ *c* в текст *T*, и, если входит, присваивающую параметрам *i* и *j* «координаты» первого вхождения этого символа: *i* – номер строки, *j* – номер позиции в этой строке;

з) процедуру *вывод*(*T*), которая выводит построчно текст *T* на экран;

и) процедуру *ввод*(*T*), считывающую из входного файла последовательность символов до первой точки и формирующую из них текст *T* (последнюю строку, если надо, дополнить пробелами).

В упражнениях 16.17 – 16.38 использовать однонаправленные списки без заглавного звена (см. рис. 12,а) или с заглавным звеном (см. рис. 12,б) при следующем их описании:

```
type ТЭ = ...;  {тип элементов списка (уточняемый,
                 если надо, в упражнениях)}
список = ↑звено;
звено = record  элем: ТЭ;  след: список end;
```

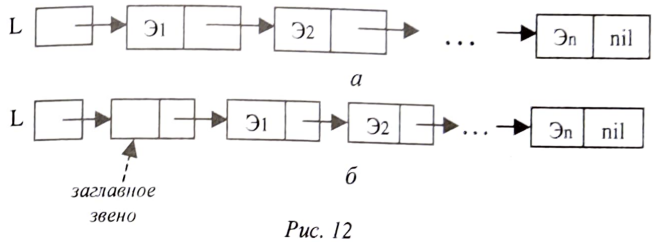


Рис. 12

При этом параметры L, L1 и L2 обозначают списки (ссылки на первое звено списков), а параметры E, E1 и E2 – данные типа ТЭ, к которым применимы операции присваивания и сравнения.

16.17*. var L, p: список; E: ТЭ; t: boolean;

Пусть переменная *p* указывает на какое-то звено списка *L* (т.е. в *p* находится ссылка на это звено). Реализовать следующие простейшие операции над списком:

а) переменной *E* присвоить элемент из звена, на которое указывает *p*;

б) переменной *t* присвоить значение *true*, если *p* указывает на последнее звено списка, и значение *false* иначе;

в) переменной *p* присвоить ссылку на следующее звено списка (т.е. «перейти» к следующему звену списка);

г) переменной *t* присвоить значение *true*, если в списке *L* более одного элемента, и значение *false* иначе (учесть, что в любой операции сначала вычисляются ее операнды и лишь затем выполняется сама операция).

16.18. Описать функцию или процедуру, которая:

а)* выдает как свое значение элемент из последнего звена непустого списка *L*;

б)* подсчитывает количество элементов в списке *L*;

в) вычисляет среднее арифметическое элементов непустого списка *L* (*ТЭ=real*);

г) находит наибольший элемент непустого списка *L*;

д)* заменяет в списке *L* все вхождения *E1* на *E2*;

е) меняет местами элементы из первого и последнего звеньев непустого списка *L*;

ж)* заменяет на *E* наименьший элемент непустого списка *L* (считать, что такой элемент единственный);

з) меняет местами наибольший и наименьший элементы непустого списка *L*, содержащего различные элементы;

и)* проверяет, упорядочены ли элементы непустого списка *L* по возрастанию;

к) проверяет, образуют ли элементы непустого списка *L* арифметическую прогрессию (*ТЭ=integer*);

л) находит сумму последнего и предпоследнего элементов списка *L*, содержащего не менее двух элементов (*ТЭ=real*);

м) подсчитывает число вхождений в непустой список *L* его наибольшего элемента.

16.19. type слово = packed array [1..10] of char;
ТЭ = слово;

Описать функцию, подсчитывающую количество слов непустого списка *L*, которые:

а) начинаются и оканчиваются одним и тем же символом;

б) начинаются с того же символа, что и следующее слово;

в) совпадают с последним словом.

16.20. type файл = file of ТЭ;
массив = array[1..50] of ТЭ;

Описать функцию, значением которой является список, построенный ею:

а)* из элементов файла *f*;

б) из чисел Фибоначчи (1, 1, 2, 3, 5, 8, …), не превосходящих натурального числа *N* (*ТЭ=integer*);

в)* из элементов массива *x* (список строить от конца);

г) из цифр (начиная со старшей) неотрицательного целого числа *N* (*ТЭ=0..9*);

д) из элементов списка *L*, но расположенных в обратном порядке.

16.21. Описать процедуру, которая по списку *L* строит два новых списка: *L1* – из положительных элементов и *L2* – из остальных элементов списка *L* (*ТЭ=real*).

16.22. Описать процедуру, которая вставляет:

а) новый элемент *E* в начало списка *L*;

б)* новый элемент *E* после первого элемента непустого списка *L*;

в)* новый элемент *E* в конец списка *L*;

г) в список *L* новый элемент *E1* за каждым вхождением элемента *E*;

д) новый элемент *E* перед *i*-м элементом списка *L*, если такой элемент есть ($i \geq 1$);

е)* в список *L* новый элемент *E1* перед первым вхождением элемента *E*, если *E* входит в *L*;

ж) в непустой список *L* пару новых элементов *E1* и *E2* перед его последним элементом.

16.23. Описать процедуру, которая удаляет:

а) из непустого списка *L* первый элемент;

б)* из списка *L* второй элемент, если такой есть;

в) *i*-й элемент из списка *L*, если такой есть ($i \geq 1$);

г) из списка *L* один элемент за каждым вхождением элемента *E*, если такой элемент есть и он отличен от *E*;

д)* из непустого списка *L* последний элемент;

е) из списка *L* первый отрицательный элемент, если такой есть (*ТЭ=integer*);

ж) из списка *L* все отрицательные элементы (*ТЭ=real*);

з) из непустого списка *L* его наибольший элемент (считать, что такой элемент единственный).

16.24. Описать процедуру *ins*(*L*,*E*), вставляющую в непустой список *L*, элементы которого упорядочены по неубыванию, новый элемент *E* так, чтобы сохранилась упорядоченность.

16.25*. Программа. Заданный во входном файле текст (за ним следует точка) вывести на экран в обратном порядке.

16.26. Программа. Дана непустая последовательность натуральных чисел, за которой следует 0. Вывести порядковые номера тех чисел последовательности, которые имеют наибольшую величину.

16.27. Программа. Дано целое $n>1$, за которым следует n вещественных чисел. Вывести эти числа в порядке неубывания.

16.28. Программа. Дана непустая последовательность слов, в каждом из которых от 1 до 8 малых латинских букв; между соседними словами – запятая, за последним словом – точка. Вывести эти слова по алфавиту.

16.29. Описать нерекурсивную процедуру или функцию, которая:

а) проверяет на равенство списки *L1* и *L2*;

б) определяет, входит ли список *L1* в список *L2* (есть ли в *L* последовательность соседних элементов, совпадающая с *L1*);

в) проверяет, есть ли в списке *L* хотя бы два одинаковых элемента;

г) находит сумму элементов между максимальным и минимальным элементами непустого списка *L*, включая и эти элементы (считать, что все элементы в *L* различны; *ТЭ=real*);

д) переносит в конец непустого списка *L* его первый элемент;

е) переносит в начало непустого списка *L* его последний элемент;

ж) добавляет в конец списка *L1* копии всех элементов списка *L2*;

з) заменяет первое вхождение списка *L1* в список *L* на копию списка *L2*, если такое вхождение есть;

и) переворачивает список *L*, т.е. изменяет ссылки в этом списке так, чтобы его элементы оказались расположенными в обратном порядке;

к) в списке *L* из каждой группы подряд идущих равных элементов оставляет только один;

л) оставляет в списке *L* только первые вхождения одинаковых элементов.

16.30. Описать рекурсивную функцию или процедуру, которая:

а)* определяет, входит ли элемент *E* в список *L*;

б) подсчитывает число вхождений элемента *E* в список *L*;

в) находит максимальный элемент непустого списка *L*;

г) заменяет в списке *L* первое вхождение *E1* на *E2*;

д) заменяет в списке *L* все вхождения *E1* на *E2*;

е) проверяет на равенство два списка *L1* и *L2*;

ж) выводит на экран в обратном порядке элементы списка *L* (*ТЭ=char*);

з)* удаляет из списка *L* первое вхождение элемента *E*, если такое есть;

и) удаляет из списка *L* все вхождения элемента *E*;

к) удаляет из непустого списка *L* последний элемент;

л)* добавляет в список *L* элемент *E1* за первым вхождением элемента *E*, если такое есть;

м) добавляет в список *L* элемент *E1* перед первым вхождением элемента *E*, если такое есть;

н) добавляет новый элемент *E* в конец списка *L*;

о) строит *L1* – копию списка *L*;

п) строит список из целых чисел от 1 до *N* в порядке убывания ($N \geq 1$);

р) строит список из целых чисел от 1 до *N* в порядке их возрастания ($N \geq 1$);

с) находит среднее арифметическое всех элементов непустого списка *L* (*ТЭ=real*).

16.31. Не используя операторы цикла и перехода, описать логическую функцию *same*(*L*) для проверки, есть ли в списке *L* хотя бы два одинаковых элемента, при следующем условии:

а) описать и использовать вспомогательную логическую функцию *memb*(*E*,*L*), проверяющую, есть ли в списке *L* элемент *E* (*ТЭ=real*);

б) использовать множество из уже просмотренных элементов списка *L* (*ТЭ=char*).

16.32. Описать процедуру, которая формирует список *L* (*ТЭ=char*), включив в него по одному разу элементы, которые:

а) входят хотя бы в один из списков *L1* и *L2*;

б) входят одновременно в оба списка *L1* и *L2*;

в) входят в список *L1*, но не входят в список *L2*;

г) входят в один из списков *L1* и *L2*, но в то же время не входят в другой из них.

16.33. type слово = ↑цепочка;
цепочка = record буква: 'a'..'z'; связь: слово end;
ТЭ = слово;

Описать функцию или процедуру, которая:

а) в списке *L* переставляет местами первое и последнее непустые слова, если в *L* есть хотя бы два непустых слова;

б)* выводит на экран текст из первых букв всех непустых слов списка *L*;

в) удаляет из непустых слов списка *L* их первые буквы;

г) выводит на экран все непустые слова списка *L*;

д) определяет количество слов в непустом списке *L*, отличных от последнего.

16.34. Описать процедуру, которая объединяет два упорядоченных по неубыванию списка *L1* и *L2* в один упорядоченный по неубыванию список:

а) построив новый список *L*;

б) меняя соответствующим образом ссылки в *L1* и *L2* и присвоив полученный список параметру *L*.

16.35. Описать процедуру *упор*(*L*), которая упорядочивает элементы непустого списка *L* по неубыванию (см. упр. 8.41):

а) методом выбора;
б) методом пузырька.

16.36. Описать процедуру *упор*(*N*,*L*), которая строит список *L* (*ТЭ*=0..9) из цифр неотрицательного целого числа *N*, упорядочив их по невозрастанию. Используя эту процедуру, решить следующую задачу: из всех целых чисел, которые можно составить из цифр заданного неотрицательного целого числа, найти наибольшее.

16.37.
```
const n=...;  {целая константа > 1}
type число = packed array [1..n] of 0..9;
     ТЭ = число;
```
Описать процедуру *упор*(*L*), упорядочивающую по неубыванию элементы непустого списка *L* с помощью следующего алгоритма (см. рис. 13, где *n* предполагается равным 2).

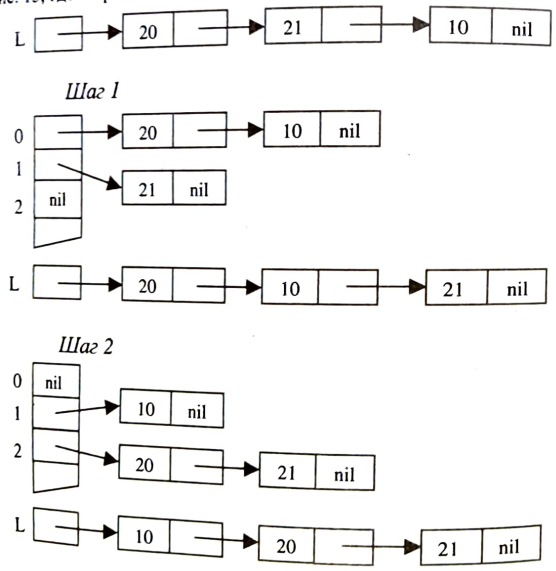


Рис.13

Создать 10 пустых подсписков (по количеству цифр), а затем,

просматривая числа исходного списка, занести в *k*-й подсписок все числа, оканчивающиеся цифрой *k*, после чего эти подсписки объединить в один список *L*, записав в последнее звено *k*-го подсписка ссылку на начало (*k*+1)-го подсписка. Далее аналогичный метод применяется по отношению к предпоследней цифре чисел (не нарушая при этом упорядоченность по последней цифре), затем – по отношению к третьей от конца цифре и т.д.

16.38. (Моделирование файла в виде списка.) Пусть последовательность элементов (типа *ТЭ*) файла представлена в виде списка, причем известны ссылка *начало* на первое звено этого списка, текущий режим работы с файлом (чтения, записи или пока еще не определенный) и ссылка *маркер* на звено с текущим элементов файла (в режиме чтения) или на последнее звено (в режиме записи):
```
type ТипРежима = (чтение, запись, неопр);
     файл = record начало, маркер: список;  режим: ТипРежима end;
```
Считая, что вначале поле *режим* любого файла имеет значение *неопр*, а другие поля не имеют значений, описать процедуры *reset1*(*f*), *read1*(*f*,*x*), *rewrite1*(*f*) и *write1*(*f*,*x*) и функцию *eof1*(*f*), реализующие при таком представлении файлов действия соответствующих стандартных процедур и функции. (В случае невозможности выполнить операцию передавать управление некоторой процедуре *Ошибка* без параметров.)

16.39. Имеется внешний файл *COURSE* из элементов типа *stud*:
```
type str = packed array[1..10] of char;
     stud = record     {информация об одном студенте}
              fn: record fam, name: str end; {фамилия, имя}
              sex: (M,W);     {пол: муж. (M), жен. (W)}
              marks: array[1..5] of 2..5;     {пять оценок}
            end;
```
Считая, что количество элементов в этом файле заранее не известно и что фамилии и имена студентов записаны большими русскими буквами, составить программу для чтения данных из этого файла и записи во внешний текстовый файл *ANS* следующей информации (о каждом студенте – в отдельной строке):

а) о каждой студентке сообщить фамилию, имя и все ее оценки;
б) о каждом отличнике сообщить фамилию, имя и пол;
в) о каждом «хорошисте» (все оценки – 4 или 5 и есть хотя бы одна 4) сообщить фамилию, пол и все оценки;

г) о каждом студенте-юноше, имеющем хотя бы одну оценку 2, сообщить фамилию, имя и все оценки;

д) составить рейтинг – список всех студентов по убыванию их средних оценок, сообщив о каждом из них фамилию, имя и среднюю оценку (с одной цифрой в дробной части).

В вариантах *а*) – *г*) в ответе фамилии упорядочить по алфавиту.

16.39. Многочлен

$P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$

с целыми коэффициентами можно представить в виде списка (см. рис. 14,*а*), элементы которого расположены по убыванию степеней одночленов и в котором при a_i=0 соответствующее звено отсутствует (на рис. 14,*б* показано представление многочлена $S(x)=52x^{40}-3x^8+x$). Нулевой многочлен представляется пустым списком.

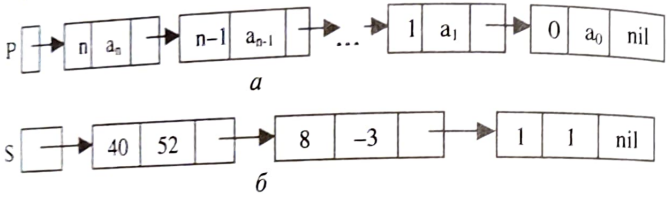


Рис. 14

Описать на Паскале тип данных, соответствующий такому представлению многочленов, и определить следующие функции и процедуры для работы с этими списками-многочленами:

а) логическую функцию *равно*(*p*,*q*), проверяющую на равенство многочлены *p* и *q*;

б) функцию *знач*(*p*,*x*), вычисляющую значение многочлена *p* в целочисленной точке *x*;

в) процедуру *диф*(*p*,*q*), которая строит многочлен *p* – производную многочлена *q*;

г) процедуру *слож*(*p*,*q*,*r*), которая строит многочлен *p* – сумму многочленов *q* и *r*;

д) процедуру *вывод*(*p*,*v*), которая выводит на экран многочлен *p* как многочлен от переменной, однобуквенное имя которой является значением символьного параметра *v*; например, для указанного выше многочлена *S* процедура *вывод*(*S*,'y') должна вывести

52y↑40–3y↑8+y

где знак ↑ означает возведение в степень (коэффициенты и степени, равные 1, не указывать);

е) процедуру *ввод*(*p*), которая считывает из входного файла безошибочную запись многочлена (ее вид см. в пункте *д*), за которой следует пробел, и формирует соответствующий список-многочлен *p*.

16.40. Программа. Дана запись многочлена (от переменной *x*) произвольной степени с целыми коэффициентами, причем его одночлены могут быть и не упорядочены по степеням *x*, а одночлены одной и той же степени могут повторяться. Возможный пример (знак ↑ означает возведение в степень):

–8x↑4–74x +8x↑4 +5–x↑3

Требуется привести подобные члены в этом многочлене, после чего вывести его на экран по убыванию степеней *x*.

16.41. Кольцевым (циклическим) списком называется однонаправленный список, в последнем звене которого вместо *nil* указывается ссылка на первое звено (см. рис. 15).

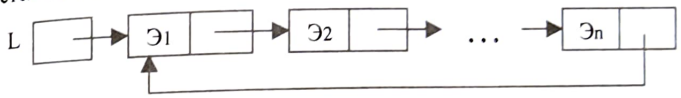


Рис. 15

Пусть *L* обозначает кольцевой список из элементов типа *ТЭ*, а *E* – величину типа *ТЭ*. Описать функцию или процедуру, которая:

а)* определяет, содержит ли список *L* ровно один элемент;

б) определяет, есть ли в списке *L* хотя бы один элемент, не равный следующему за ним (по кругу) элементу;

в) выводит на экран следующую таблицу из элементов $Э_i$ непустого списка *L* длины *n* (*ТЭ*=*char*):

$Э_1\ Э_2 \ldots Э_{n-1}\ Э_n$

$Э_2\ Э_3 \ldots Э_n\ Э_1$

. . .

$Э_n\ Э_1 \ldots Э_{n-2}\ Э_{n-1}$

г) строит кольцевой список *L* по однонаправленному списку *L1*;

д) строит однонаправленный список *L1* по кольцевому списку *L*;

е)* удаляет из непустого списка *L* последний элемент;

ж) удаляет из непустого списка *L* первый элемент (учесть, что должна быть изменена ссылка в последнем звене);

з) добавляет в конец списка *L* новый элемент *E*;

и) добавляет в начало списка *L* новый элемент *E* (учесть, что должна быть изменена ссылка в последнем звене).

16.42. («Считалка».) n ребят располагаются по кругу. Начав отсчет от первого, удаляют каждого k-го, смыкая круг после каждого удаления. Определить порядок удаления ребят из круга.

Решение этой задачи описать в виде программы (ее исходные данные – натуральные числа n и k), которая должна вывести на экран номера ребят в порядке их удаления из круга.

16.43. Двунаправленным списком называется список, в каждом звене которого хранятся ссылки на следующее и предыдущее звенья (на рис. 16 показан двунаправленный список с заглавным звеном, в котором находятся ссылки на первое и последнее звенья списка). Возможное описание такого списка:

```
type ТЭ2 = ...; {тип элементов списка}
    список2 = ↑звено2;
    звено2 = record элем: ТЭ2;  пред, след: список2 end;
```

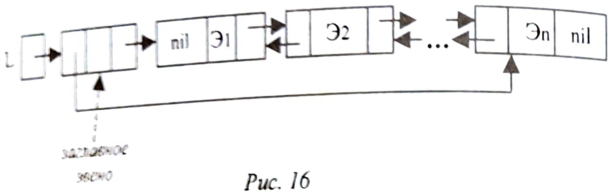


Рис. 16

Пусть L обозначает двунаправленный список (с заглавным звеном) типа *список2*, а E – величину типа *ТЭ2*. Описать функцию или процедуру, которая:

а)* создает список из одного элемента E;

б)* выводит на экран в обратном порядке элементы списка L (*ТЭ2=char*);

в) выводит на экран сначала все отрицательные элементы списка L в прямом порядке, а затем – все остальные элементы в обратном порядке (*ТЭ2=integer*);

г) определяет, симметричен ли список L;

д) подсчитывает количество элементов списка L, у которых равные «соседи»;

е) в списке L переставляет в обратном порядке все элементы между первым и последним вхождениями элемента E, если E входит в L не менее двух раз;

ж) добавляет в начало списка L новый элемент E;

з) добавляет в конец списка L новый элемент E;

и) добавляет элемент E между всеми равными соседними элементами списка L;

к) строит двунаправленный список L по однонаправленному списку $L1$;

л) удаляет из непустого списка L первый элемент;

м) удаляет из непустого списка L последний элемент;

н) из списка L удаляет все элементы, у которых (в исходном списке) одинаковые «соседи».

16.44. Программа. Дан текст, оканчивающийся точкой. Среди символов этого текста особую роль играет знак #, появление которого в тексте означает отмену предыдущего символа текста; k знаков # подряд отменяют k предыдущих символов (если такие есть). Вывести на экран данный текст, исправленный с учетом такой роли знака # (например, текст ХЭ#E##HELO#LO должен быть вывести в виде HELLO).

17. ОЧЕРЕДИ, СТЕКИ, ДВОИЧНЫЕ ДЕРЕВЬЯ

17.1. Для работы с очередью, т.е. последовательностью элементов, в которую элементы всегда добавляются в конец, а удаляются из начала («первым пришел – первым вышел»), нужны обычно следующие операции:

ОЧИСТОЧ(Q) – создать пустую очередь Q (очистить очередь);

ПУСТОЧ(Q) – проверить, является ли очередь Q пустой;

ВОЧЕРЕДЬ(Q,x) – добавить в конец очереди Q элемент x;

ИЗОЧЕРЕДИ(Q,x) – удалить из очереди Q первый элемент, присвоив его параметру x.

Требуется для каждого из указанных ниже представлений очереди описать на языке Паскаль соответствующий тип *очередь*, считая, что все элементы очереди имеют некоторый тип *ТЭО*, и реализовать в виде процедур или функций перечисленные операции над очередью (если операция по тем или иным причинам не может быть выполнена, следует передать управление некоторой процедуре *ОШИБКА*(k), считая ее уже описанной, где k – номер ошибки: 1 – переполнение очереди, 2 – исчерпание очереди).

Представление очереди (n – целая константа >1):

а)* для каждой очереди отводится свой массив из n компонент типа *ТЭО*, в котором элементы очереди занимают группу соседних

компонент, индексы первой и последней из которых запоминаются (см. рис. 17,*а*); когда очередь достигает правого края массива, все ее элементы сдвигаются к левому краю;

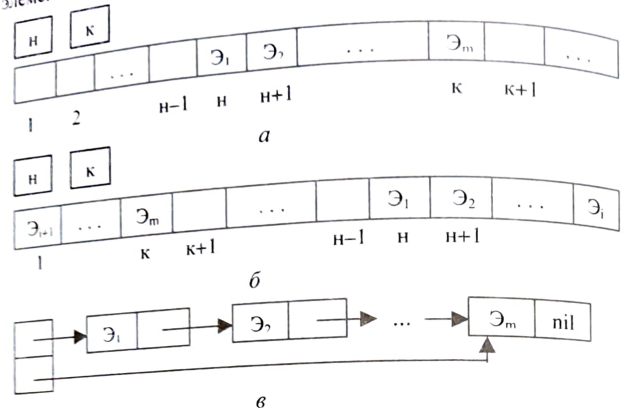


Рис. 17

б) аналогичное представление, но массив как бы склеивается в кольцо, поэтому при достижении очередью правого края массива новые элементы записываются в начало массива (см. рис. 17,*б*);

в) для каждой очереди создается свой однонаправленный список из элементов типа *ТЭО*, при этом запоминаются ссылки на первое и последнее звенья списка (см. рис. 17,*в*).

17.2. Используя очередь (считать уже описанными тип *очередь* при подходящем типе *ТЭО*, функцию *ПУСТОЧ* и процедуры *ОЧИСТОЧ*, *ВОЧЕРЕДЬ* и *ИЗОЧЕРЕДИ* – см. упр. 17.1), решить следующую задачу (решение описать в виде процедуры).

а)* type FR = file of real;

За один просмотр файла *f* типа *FR* вывести на экран его элементы в следующем порядке: сначала – все числа, меньшие *a*, потом – все числа из отрезка [*a*,*b*], а затем – все остальные числа, сохраняя исходный взаимный порядок в каждой из этих трех групп чисел (*a* и *b* – заданные числа, *a*<*b*).

б) Содержимое текстового файла *f*, разделенное на строки, переписать в текстовый файл *g*, перенося при этом в конец каждой строки все входящие в нее цифры (с сохранением исходного взаимного порядка как среди цифр, так и среди остальных символов строки).

в) type имя = (Анна, …, Яков);
дети = packed array [имя, имя] of boolean;
потомки = file of имя;

Считая заданными имя *И* и массив *Д* типа *дети* (*Д*[*x*,*y*]=*true*, если человек по имени *y* является ребенком человека по имени *x*), записать в файл *П* типа *потомки* имена всех потомков человека с именем *И* в следующем порядке: сначала – имена всех его детей, потом – всех его внуков, затем – всех правнуков и т.д.

г) Во входном файле задана последовательность из равного количества положительных и отрицательных целых чисел, за которой следует 0. Ввести эти числа и вывести их, чередуя положительные числа (на нечетных местах) с отрицательными (на четных местах), причем исходный взаимный порядок как среди положительных, так и среди отрицательных чисел должен быть сохранен.

17.3. Для работы со стеком, т.е. последовательностью элементов, в которую элементы всегда добавляются в конец и удаляются из конца («последним пришел – первым вышел»), нужны обычно следующие операции:

ОЧИСТЕК(*S*) – создать пустой стек *S* (очистить стек);
ПУСТЕК(*S*) – проверить, является ли стек *S* пустым;
ВСТЕК(*S*,*x*) – добавить в конец стека *S* элемент *x*;
ИЗСТЕКА(*S*,*x*) – удалить из стека *S* последний элемент, присвоив его параметру *x*.

Требуется для каждого из указанных ниже представлений стека описать на языке Паскаль соответствующий тип *стек*, считая, что все элементы стека имеют некоторый тип *ТЭС*, и реализовать в виде процедур или функций перечисленные операции над стеком (если операция по тем или иным причинам невыполнима, следует передать управление некоторой процедуре *ОШИБКА*(*k*), считая ее уже описанной, где *k* – номер ошибки: 1 – переполнение стека, 2 – исчерпание стека).

Представление стека (*n* – целая константа >1):

а) для каждого стека отводится свой массив из *n* компонент типа *ТЭС*, в начале которого располагаются элементы стека, при этом запоминается индекс компоненты массива, занятой последним элементом стека (см. рис. 18,*а*);

б) для каждого стека создается свой однонаправленный спи-

сок, в котором элементы стека располагаются в обратном порядке (см. рис. 18,*б*).

17.4. Используя стек (считать уже описанными тип *стек* при *ТЭС=char*, функцию *ПУСТЕК* и процедуры *ОЧИСТЕК*, *ВСТЕК* и *ИЗСТЕКА* – см. упр. 17.3), решить следующую задачу (решение описать в виде процедуры или функции).

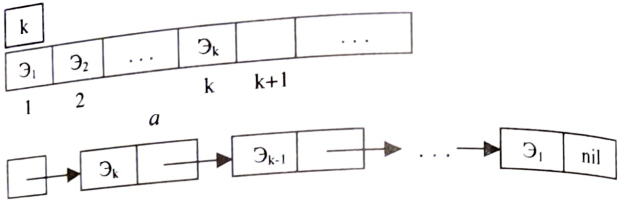


Рис. 18

а) Вывести на экран содержимое текстового файла *t*, выписывая символы каждой его строки в обратном порядке.

б) type FC = file of char;
Проверить, сбалансировано ли содержимое файла *t* типа *FC* относительно круглых, квадратных и фигурных скобок.

в)* Во входном файле записана формула, определяемая следующим образом:

<формула> ::= <цифра> | M(<формула>,<формула>) |
m(<формула>,<формула>)
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

где *M* обозначает максимум, а *m* – минимум; за формулой следует точка. Вычислить (как целое число) значение данной формулы (например, M(5,m(6,8)) → 6).

г) Во входном файле записана формула, определяемая следующим образом:

<формула> ::= <цифра> | (<формула> <знак> <формула>)
<знак> ::= + | – | *
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

За формулой следует точка. Вычислить значение этой формулы (например, ((2–4)*6) → –12).

д) Во входном файле записано логическое выражение (ЛВ), определяемое следующим образом:

<ЛВ> ::= true | false | (¬ <ЛВ>) | (<ЛВ> ∧ <ЛВ>) | (<ЛВ> ∨ <ЛВ>)

где знаки ¬, ∧ и ∨ обозначают соответственно отрицание, конъюнкцию и дизъюнкцию; за выражением следует точка. Вычислить (как *boolean*) значение этого выражения.

17.5. Используя очередь и/или стек (считать уже описанными их типы и операции над ними – см. упр. 17.1 и 17.3), решить следующую задачу (решение описать в виде процедуры).

а) type FI = file of integer;
Пусть *a* и *b* – заданные целые числа, *a*<*b*. Вывести на экран элементы файла *f* типа *FI* следующим образом: сначала – в обратном порядке все числа, меньшие *a*, потом – в прямом порядке все числа из отрезка [*a*,*b*], а затем – в обратном порядке все остальные числа.

б) type FC = file of char;
В файле *t* типа *FC* записан текст, сбалансированный по круглым скобкам. Требуется для каждой пары соответствующих открывающей и закрывающей скобок вывести на экран номера их позиций в тексте, упорядочив пары номеров в порядке возрастания номеров позиций закрывающих скобок. Например, для текста A+(45–F(X)*(B–C)) надо вывести:

8 10; 12 16; 3 17

в) Решить предыдущую задачу, но пары номеров выводить в порядке возрастания номеров позиции открывающих скобок. Например, для текста A+(45–F(X)*(B–C)) надо вывести:

3 17; 8 10; 12 16

г) Не используя рекурсию, решить задачу о ханойских башнях (см. упр. 12.33).

17.6. Под «выражением» будем понимать конструкцию, определяемую следующим образом:

<выражение> ::= <терм> | <терм> <знак ±> <выражение>
<знак ±> ::= + | –
<терм> ::= <множитель> | <множитель>*<терм>
<множитель> ::= <операнд> |
<множитель> ↑ <операнд> | (<выражение>)
<операнд> ::= <число> | <переменная>
<число> ::= <цифра>
<переменная> ::= <буква>

где знак ↑ обозначает операцию возведения в степень.

Постфиксной формой записи выражения $a\Delta b$ называется запись, в которой знак операции размещен за операндами: $ab\Delta$. Примеры (в постфиксной записи скобки не указываются):

a–b → ab–
a*b+c → ab*c+
a*(b+c) → abc+*
a+b↑c↑d*e → abc↑d↑e*+

а) Описать функцию *value*(*postfix*), которая вычисляет как целое число значение выражения (без переменных), записанного в постфиксной форме в текстовом файле *postfix*.

Использовать следующий алгоритм вычисления. Выражение просматривается слева направо. Если встречается операнд (число), то он заносится в стек, а если встречается знак операции, то из стека извлекаются два последних элемента (это операнды данной операции), над ними выполняется операция и ее результат записывается в стек. В конце в стеке останется только одно число – значение всего выражения.

б) Описать процедуру *translate*(*infix*, *postfix*), которая переводит выражение, записанное в обычной (инфиксной) форме в текстовом файле *infix*, в постфиксную форму и в таком виде записывает его в текстовый файл *postfix*.

Использовать следующий алгоритм перевода. В стек записывается открывающая скобка, и выражение просматривается слева направо. Если встречается операнд (число или переменная), то он сразу переносится в файл *postfix*. Если встречается открывающая скобка, то она заносится в стек, а если встречается закрывающая скобка, то из стека извлекаются находящиеся там знаки операций до ближайшей открывающей скобки, которая также удаляется из стека, и все эти знаки (в порядке их извлечения) записываются в файл *postfix*. Когда же встречается знак операции, то из стека извлекаются (до ближайшей скобки, которая сохраняется в стеке) знаки операций, старшинство которых больше или равно старшинству данной операции, и они записываются в файл *postfix*, после чего рассматриваемый знак заносится в стек. В заключение выполняются такие же действия, как если бы встретилась закрывающая скобка.

в) Описать (нерекурсивную) процедуру *infixprint*(*postfix*), которая выводит на экран в обычной (инфиксной) форме выражение, записанное в постфиксной форме в текстовом файле *postfix*. (Лишние скобки желательно не выводить.)

В последующих упражнениях использовать двоичные деревья (см. рис. 19) при следующем их описании:

```
type ТЭД = ...;   {тип элементов дерева (уточняемый, если
                   надо, в упражнениях)}
дерево = ↑вершина;
вершина = record  элем: ТЭД;  лев, прав: дерево  end;
```

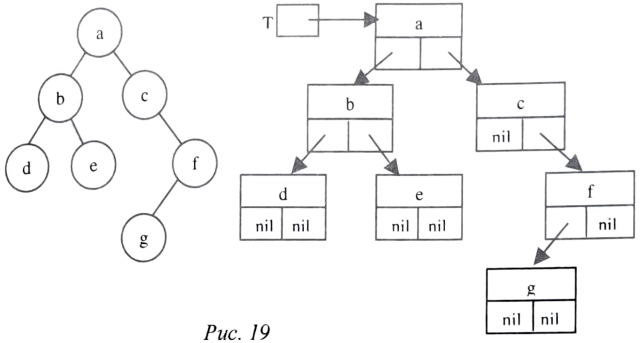


Рис. 19

В этих упражнениях T, T1 и T2 обозначают двоичные деревья (ссылки на корень деревьев), а E – величину типа ТЭД, к данным которого применимы операции присваивания и сравнения. Корень считается вершиной 1-го уровня, его дочерние вершины – 2-го уровня и т.д. Листом называется вершина, из которой не выходит ни одной ветви.

17.7. Используя очередь или стек (считать уже описанными их типы и операции над ними – см. упр. 17.1 и 17.3), описать процедуру или функцию, которая:

а) присваивает параметру *E* элемент из самого левого листа непустого дерева *T*;

б)* определяет число вхождений элемента *E* в дерево *T*;

в) вычисляет среднее арифметическое всех элементов непустого дерева *T* (*ТЭД=real*);

г) заменяет в дереве *T* все отрицательные элементы на их абсолютные величины (*ТЭД=real*);

д) меняет местами максимальный и минимальный элементы непустого дерева *T*, все элементы которого различны;

е) выводит на экран элементы из всех листьев дерева *T*

(*ТЭД=char*);

ж)* выводит на экран все элементы дерева T по уровням: сначала – из корня дерева, затем (слева направо) – из вершин, дочерних по отношению к корню, затем (также слева направо) – из вершин, дочерних по отношению к этим вершинам, и т.д. (*ТЭД=integer*);

з) подсчитывает в непустом дереве T число ветвей на пути от корня до ближайшей вершины с элементом E; если E не входит в T, за ответ принять –1;

и) подсчитывает число вершин на n-ом уровне дерева T ($n \geq 1$);

к) определяет номер уровня дерева T, на котором больше всего вершин.

17.8. Описать рекурсивную функцию или процедуру, которая:

а) определяет, входит ли элемент E в дерево T;

б)* определяет число вхождений элемента E в дерево T;

в) вычисляет сумму элементов непустого дерева (*ТЭД=real*);

г) находит величину наибольшего элемента непустого дерева T;

д) подсчитывает число листьев в дереве T;

е) выводит на экран элементы из всех листьев (слева направо) дерева T (*ТЭД=char*);

ж) подсчитывает число вершин дерева T, из которых выходят ровно две ветви;

з) подсчитывает число вершин дерева T, из которых выходит ровно одна ветвь;

и)* определяет высоту дерева T – число вершин на самом длинном из путей от корня дерева до листьев (пустое дерево имеет высоту 0);

к) подсчитывает число вершин на n-ом уровне дерева T ($n \geq 1$);

л) определяет, есть ли хотя бы один лист на n-м уровне дерева T ($n \geq 1$).

17.9. Логическую функцию *eq*($T1$,$T2$), проверяющую на равенство деревья $T1$ и $T2$, описать:

а) нерекурсивно (используя стек или очередь);

б) рекурсивно.

17.10. Описать логическую функцию *same*(T), определяющую, есть ли в дереве T хотя бы два одинаковых элемента (*ТЭД=char*). (*Рекомендация*: использовать множество уже просмотренных элементов.)

17.11*. Описать процедуру *copy*(T,$T1$), которая строит $T1$ – копию дерева T.

17.12. Описать процедуру *create*(T,n), где n – натуральное число, которая строит T – дерево, показанное на рис. 20.

17.13. Описать процедуру *del*(T), которая уничтожает дерево T, освобождая память, занятую его вершинами, и присваивая *nil* параметру T.

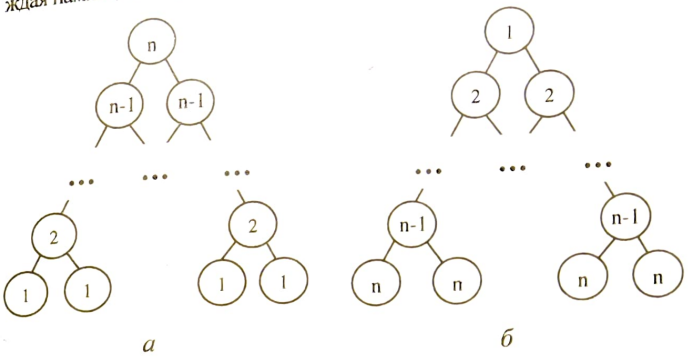


Рис. 20

17.14. Не используя операторы цикла и перехода, описать логическую функцию *path*(T,S), определяющую, есть ли в непустом дереве T из натуральных чисел (*ТЭД=1..maxint*) такой путь от корня до какого-нибудь листа, в котором:

а)* сумма всех его элементов равна целому числу S;

б) элементы возрастают (по направлению к листу);

в) есть хотя бы два одинаковых элемента;

г) нет двух одинаковых элементов.

17.15. Формулу вида

<формула> ::= <терм> | (<формула><знак><формула>)

<знак> ::= + | – | *

где *терм* – это цифра или малая латинская буква, можно представить в виде двоичного дерева («дерева-формулы») с *ТЭД=char* согласно следующим правилам: формула из одного терма представляется деревом из одной вершины с этим термом, а формула вида ($\varphi_1 \Delta \varphi_2$) – деревом, в корне которого записан знак Δ, а левое и правое поддеревья – это соответствующие представления формул φ_1 и φ_2. (На рис. 21 показано дерево-формула, соответствующее формуле (x*(y–1)).)

Описать рекурсивную функцию или процедуру, которая:

а)* вычисляет (как целое число) значение дерева-формулы T, в которой все термы являются цифрами;

б) проверяет, является ли двоичное дерево T деревом-формулой;

в) по формуле из входного файла строит соответствующее дерево-формулу T;

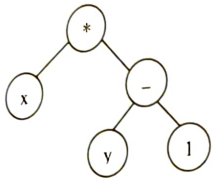


Рис. 21

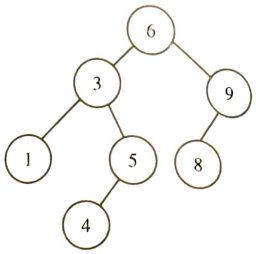


Рис. 22

г) выводит дерево-формулу T в виде соответствующей формулы (со скобками);

д) выводит дерево-формулу T в постфиксной записи, в которой знаки операций ставятся после их операндов и скобки не указываются (например: $(x*(y–1)) \rightarrow xy1–*$);

е) выводит дерево-формулу T в префиксной записи, в которой знаки операций ставятся до их операндов и скобки не указываются (например: $(x*(y–1)) \rightarrow *–xy1$).

17.16. Пусть T – дерево-формула (см. упр. 17.15). Описать процедуру, которая:

а)* упрощает это дерево-формулу, заменяя в нем все поддеревья, соответствующие формулам (φ+0), (0+φ), (φ–0), (φ*1) и (1*φ), на поддерево, соответствующее формуле φ, а поддеревья, соответствующие формулам (φ*0) и (0*φ), – на вершину с 0;

б) преобразует это дерево-формулу, заменяя в нем все поддеревья, соответствующие формулам $((\varphi_1 \pm \varphi_2)*\varphi_3)$ и $(\varphi_1*(\varphi_2 \pm \varphi_3))$, на поддеревья, соответствующие $((\varphi_1*\varphi_3) \pm (\varphi_2*\varphi_3))$ и $((\varphi_1*\varphi_2) \pm (\varphi_1*\varphi_3))$;

в) строит дерево-формулу $T1$ – производную дерева-формулы T по переменной, имя которой задано как параметр.

17.17. Деревом поиска (деревом сравнения) называется двоичное дерево, в котором слева от каждой вершины находятся вершины с элементами, меньшими элемента из этой вершины, а справа – с

бо́льшими элементами (элементы дерева поиска обязательно должны быть различны); пример такого дерева показан на рис. 22.

Используя типы *ТЭД* и *дерево* (см. выше), а также типы *TF* и *TA*:

```
const N=...; {N>1}
type TF = file of ТЭД;
     TA = array[1..N] of ТЭД;
```

и считая E параметром типа *ТЭД*, описать функцию или процедуру, которая:

а)* проверяет, входит ли элемент E в дерево поиска T;

б) находит наименьший элемент непустого дерева поиска T (лишних сравнений не делать);

в) определяет, есть ли в дереве поиска T хотя бы один элемент, меньший величины E;

г) записывает в файл f типа *TF* все элементы дерева поиска T в порядке их возрастания;

д) записывает в файл f типа *TF* все элементы дерева поиска T в порядке их убывания;

е) выводит на экран в порядке возрастания все элементы дерева поиска T, меньшие величины E (*ТЭД=integer*);

ж) добавляет к дереву поиска T новый элемент E (если его не было в T) так, чтобы T осталось деревом поиска;

з) из элементов файла f типа *TF*, которые различны, строит дерево поиска T;

и) из элементов массива X типа *TA*, которые упорядочены по возрастанию, строит дерево поиска T наименьшей высоты;

к) проверяет, является ли двоичное дерево T деревом поиска;

л) удаляет из непустого дерева поиска T наименьший элемент так, чтобы T осталось деревом поиска;

м) если в дерево поиска T входит элемент E, удаляет этот элемент так, чтобы T осталось деревом поиска;

н) объединяет два дерева поиска $T1$ и $T2$ в одно новое дерево поиска T;

о) строит новое дерево поиска $T1$ из тех элементов дерева поиска T, которые меньше величины E;

п) меняя только ссылки в дереве поиска T и не создавая новых вершин, выделяет из T такое дерево поиска, в которое входят элементы дерева T, меньшие величины E.

17.18. Программа. Во внешнем текстовом файле *PROG* записана (без ошибок) некоторая программа на языке Паскаль. Известно, что в

этой программе каждый идентификатор (служебное слово или имя) содержит не более 9 латинских букв и/или цифр. Вывести на экран в алфавитном порядке все различные идентификаторы этой программы, указав для каждого из них число его вхождений в текст программы. (Учесть, что в идентификаторах одноименные большие и малые буквы отождествляются, что внутри символьных значений, строк-констант и комментариев последовательности из букв и цифр не являются идентификаторами и что в записи вещественных чисел может встретиться как буква *E*, так и буква *e*.)

Для хранения идентификаторов использовать дерево поиска (см. упр. 17.17), элементами которого являются пары – идентификатор и число его вхождений в текст программы.

17.19. Решить предыдущую задачу, но вместе с каждым идентификатором выводить в возрастающем порядке номера всех строк текста программы, в которых он встречается.

18. СМЕШАННЫЕ ЗАДАЧИ

18.1. Определить, всегда ли верно в языке Паскаль следующее утверждение. Если нет, привести контрпример или доводы против.

а) Любое (нестандартное) имя можно использовать только после того, как оно описано.

б) Одним и тем же именем нельзя обозначать в программе разные объекты.

в) При описании никакая переменная не получает начального значения.

г) В любой операции сначала вычисляются ее операнды и лишь затем выполняется сама операция.

д) Каждый конструктор типа определяет новый тип, отличный от всех других типов.

е) Областью видимости (нестандартного) имени является весь блок, в котором оно описано, за исключением тех вложенных блоков, где описано такое же имя, и тех записей, где одно из полей обозначено тем же именем.

ж) Областью видимости стандартного имени является вся программа.

з) Правильная программа может начинаться со следующих строк:

```
program P (input, output);
```



```
type real=real;
```

и) Тип (*integer* или *real*) числа определяется по его написанию, а не по величине.

к) Вещественные числа представляются в памяти компьютера приближенно, поэтому сравнение этих чисел на равенство бессмысленно.

л) Везде, где можно указывать вещественное число, можно указать и целое число.

м) Везде, где можно указывать вещественную переменную, можно указать и целую переменную.

н) При любом логическом выражении *a* значение *a or* (*not a*) всегда равно *true*.

о) В стандартный тип *char* входит ровно 256 символов.

п) В записи все поля не могут быть одного и того же типа.

р) Никакая величина не может принадлежать двум разным типам (если не учитывать ограниченные типы).

с) Операция + определена только для чисел.

т) Операция = неприменима ни к каким данным сложных типов.

у) Операция < неприменима ни к каким данным сложных типов.

ф) Операция присваивания определена для любого типа данных.

х) В операторе присваивания *v*:=*e* типы переменной *v* и выражения *e* должны быть одинаковыми.

ц) Если к функции обращаться несколько раз с одними и теми же фактическими параметрами, то она всякий раз будет выдавать одно и то же значение.

ч) Если меткой помечен какой-то оператор, то в программе обязательно должен быть оператор перехода на эту метку.

ш) Параметром стандартной процедуры ввода *read* не может быть массив;

щ) Параметром стандартной процедуры вывода *write* не может быть массив.

18.2. Определить, может ли в правильной Паскаль-программе встретиться (вне строк и комментариев) следующий оператор присваивания. Если да, то привести подходящие описания имен, входящих в него, а иначе показать, что ни при каких описаниях такой оператор не может быть правильным.

а) x.y:=x.x+y.y;	б) x.p[p]:=[p];	в) y:=x[y]+y/2;
г) r:=r in [p+q];	д) a:=[a=b];	е)a:=maxint+['4', false];
ж) integer:=real;	з) x:=x↑;	и) x:=x[1];
к) x:=x[1]↑;	л) x:=x(1);	м) x:=x(1)↑;

н) x:=x↑.x;
о) x:=x.↑x;
п) x:=x.x↑;
р) a↑↑:=a↑=a;
с) q:=(p<=r)=p↑;
т) y:=x[y↑]

18.3. Программа. Для произвольных вещественных чисел a, b, c, d и e найти все решения уравнения

$$\sqrt{ax^2+bx+c} = dx+e$$

18.4. Программа. Внутри прямоугольной области со сторонами a и b (см. рис. 23) с равномерной скоростью v движется материальная точка, которая отражается от границ области по закону «угол падения равен углу отражения». В нулевой момент времени точка имела координаты (0,0) и двигалась под углом φ градусов ($0<\varphi<90$) относительно оси OX. Определить координаты точки в момент времени t.

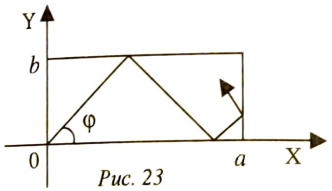


Рис. 23

18.5. Программа. Даны натуральные числа P, Q и n. Не используя вещественную арифметику (ни один промежуточный результат не должен быть вещественным), вывести значение дроби P/Q в виде вещественного числа с n цифрами в дробной части (вычислять $(n+1)$-ю цифру и округлять ее не надо). Например, при $P=169$, $Q=14$ и $n=5$ надо вывести 12.07142.

18.6. Программа. Найти период десятичной дроби P/Q, где P и Q – заданные натуральные числа ($P<Q$).

18.7. Программа. Определить число, полученное выбрасыванием всех цифр 3 из записи заданного неотрицательного целого числа.

18.8. Программа. Определить наибольшее из чисел, которые получаются из записи заданного целого числа (≥10) отбрасыванием одной из его цифр. (Например, 7809 → 809.)

18.9. Программа. Для всех целых чисел от 5 до *maxint* проверить следующее утверждение: любое простое число вида $4k+1$ можно представить в виде суммы двух квадратов ($5=1^2+2^2$, $13=2^2+3^2$, $17=1^2+4^2$ и т.д.).

18.10. Программа. Дано натуральное число k. Определить k-ю цифру последовательности:

а) 12345678910111213..., в которой выписаны подряд все натуральные числа;

б) 149162536..., в которой выписаны подряд квадраты всех натуральных чисел;

в) 1123581321..., в которой выписаны подряд все числа Фибоначчи.

18.11. Написать программу следующей игры с человеком: человек загадывает некоторое число от 1 до 999, а программа должна не более чем за 10 вопросов типа «да-нет» (например, «Равно ли Ваше число 123?», «Ваше число меньше 666?» и т.п.) угадать это число.

18.12. Программа. Ввести целое число от 1 до 999, записанное римскими цифрами (за числом следует пробел), и вывести его в обычной десятичной записи. (Например, DCCXLVI → 746, CMXCIX → 999.)

18.13. Программа. Дано целое число от 1 до 999. Вывести словесное обозначение этого числа. (Например, 1→ *один*, 125 → *сто двадцать пять*.)

18.14. Программа. Дана дата (число, месяц, год) некоторого дня нашей эры по старому стилю (по юлианскому календарю). Определить дату этого дня по новому стилю (по григорианскому календарю).

18.15. В старояпонском календаре был принят 60-летний цикл, состоявший из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. (1984 год – *год зеленой крысы* – был началом очередного цикла).

Написать программу, которая вводит номер некоторого года нашей эры и выводит его название по старояпонскому календарю. (Слова в названии должны быть грамматически согласованы.)

18.16. Астрологи делят год на 12 периодов и каждому из них ставят в соответствие один из знаков Зодиака:

20.1–18.2 – Водолей	23.7–22.8 – Лев
19.2–20.3 – Рыбы	23.8–22.9 – Дева
21.3–19.4 – Овен	23.9–22.10 – Весы
20.4–20.5 – Телец	23.10–22.11 – Скорпион
21.5–21.6 – Близнецы	23.11–21.12 – Стрелец
22.6–22.7 – Рак	22.12–19.1 – Козерог

Написать программу, которая вводит дату некоторого дня года (типа «27

мая») и выводит на экран название соответствующего знака Зодиака.

18.17. Программа. Для заданного целого числа N ($2 \le N \le 16$) вывести на экран таблицу умножения в системе счисления с основанием N.

18.18. Программа. Вывести на экран все цифры десятичной записи чисел:
а) 2^{500}; б) 500!; в) 1!+2!+3!+ … +100!

(*Рекомендация*: представить «длинные» натуральные числа в виде массивов из цифр или из групп соседних цифр и реализовать нужные операции над ними.)

18.19. Программа. Даны вещественные числа $a_0, a_1, \ldots, a_n$ ($n \le 15$). Найти коэффициенты многочлена $(x-a_0)(x-a_1)\ldots(x-a_n)$.

18.20. Программа. По заданным коэффициентам многочлена n-й степени и многочлена m-й степени ($n=15$, $m=8$) определить коэффициенты произведения этих многочленов.

18.21. Программа. По заданным коэффициентам многочлена $P(x)$ n-й степени и многочлена $Q(x)$ m-й степени ($n=10$, $m=6$) определить коэффициенты многочлена $P(Q(x))$.

18.22. Программа. Дано целое $n>2$. Определить коэффициенты n-го многочлена Чебышева $T_n(x)$, определяемого формулами

$T_0(x)=1$; $T_1(x)=x$; $T_k(x)=2xT_{k-1}(x)-T_{k-2}(x)$ при $k \ge 2$.

18.23. Программа. Даны целое n от 2 до 20 и вещественное $\varepsilon>0$. Найти с точностью ε все корни n-го многочлена Чебышева $T_n(x)$ (см. предыдущую задачу).

(*Замечание*: многочлен $T_k(x)$ имеет k различных корней в интервале (–1, 1); если $x_1<x_2<\ldots<x_k$ – корни многочлена $T_k(x)$, то многочлен $T_{k+1}(x)$ имеет по одному корню на каждом из интервалов $(-1, x_1)$, (x_1, x_2), ..., $(x_k, 1)$.)

18.24. Программа. По заданным коэффициентам $A_{i,j}$ и правым частям b_i решить систему линейных уравнений

$$\sum_{j=1}^{n} A_{i\,j}\, x_i = b_j \qquad (i = 1,\ 2,\ \ldots,\ n;\ n = 10)$$

считая, что ее определитель отличен от 0. (*Рекомендация*: систему предварительно привести к «треугольному» виду.)

18.25. Программа. Даны координаты n векторов n-мерного линейного пространства ($n=7$). Определить, являются ли они линейно независимыми.

18.26 Программа. Дана квадратная матрица n-го порядка ($n=6$). Найти матрицу, обратную ей, или установить, что такой не существует. (*Замечание*: если элементарными преобразованиями строк привести заданную матрицу к единичной, то этими же преобразованиями единичная матрица будет приведена к искомой обратной матрице.)

18.27. Программа. Дана последовательность из не менее чем двух различных натуральных чисел, за которой следует 0. Вывести на экран в обратном порядке все числа между наибольшим и наименьшим числами этой последовательности.

18.28. Программа. Дана непустая последовательность непустых слов из букв; между соседними словами – запятая, за последним словом – точка. Вывести все слова максимальной длины.

18.29. Программа. Дана непустая последовательность непустых слов из малых латинских букв; между словами – пробел, за последним словом – точка. Вывести эти слова по алфавиту, указав для каждого из них число его вхождений в эту последовательность.

18.30. Программа. Для ввода дана формула, записанная в инфиксной форме:

<формула>::=<операнд> | (<формула> <знак> <формула>)
<знак> ::= + | – | * | /

где *операнд* – цифра или буква. Вывести эту формулу в префиксной форме, в которой знаки операций ставятся перед операндами и не используются скобки, например:

(a*(b+c)/(a–d)) → /*a+bc–ad

18.31. Промоделировать выполнение любого заданного нормального алгоритма Маркова над любым заданным входным словом (см., например, [8]).

18.32 Промоделировать работу машины Тьюринга (см., например, [9]).

18.33. Программа. Даны две квадратные целые матрицы n-го порядка ($n=8$). Определить, можно ли перестановкой строк в одной из них получить другую матрицу. (Учесть, что в каждой матрице могут быть одинаковые строки).

18.34. Программа. Дан массив из n натуральных чисел ($n=20$) и натуральное число S. Требуется заменить нулем минимальное количество элементов в этом массиве, чтобы сумма его элементов стала меньше S. Вывести преобразованный массив.

18.35. Программа («задача о рюкзаке»). О каждом из n предметов ($n=30$) известны его вес и стоимость. Определить, какие предметы надо положить в рюкзак, чтобы их общий вес не превышал заданной

границы, а стоимость было максимальной.

18.36. Программа. Дано n натуральных чисел $x_1, x_2, \ldots, x_n$ (n=10) и неотрицательное целое число S. Определить, можно ли между числами x_i так расставить знаки +, – и *, чтобы получившиеся выражение имело значение S. (Скобки не использовать, минус перед первым числом не ставить.) Если можно, тогда вывести это выражение.

18.37. Программа. Даны координаты концов n отрезков $[a_i, b_i]$ и еще одного отрезка $[c, d]$ на одной и той же прямой ($a_i<b_i$, $c<d$, n=30). Определить, покрывает ли объединение отрезков $[a_i, b_i]$ отрезок $[c, d]$.

18.38. Программа. Даны координаты концов n отрезков на одной и той же прямой (n=30). Найти точку (любую из возможных), которая принадлежит наибольшему количеству этих отрезков. Вывести это количество и координату данной точки.

18.39. Программа. Даны координаты n точек на плоскости (n=20). Найти координаты центра и радиус минимального круга (т.е. с минимальным радиусом), покрывающего все эти точки.

18.40. Программа. Даны координаты центров и радиусы n кругов на плоскости (n=20). Определить, существует ли хотя бы одна точка, принадлежащая одновременно всем этим кругам, и, если существует, вывести ее координаты.

ОТВЕТЫ И РЕШЕНИЯ

1. ЧИСЛОВЫЕ ТИПЫ. ОПЕРАТОР ПРИСВАИВАНИЯ

1.1. Неправильные: а), б), д), е), з)

1.4. а) 120; б) 64; в) –10000; г) 23 (или 023, или 0023, …)

1.5. а) 3; б) 2; в) 5; г) 0; д) –3; е) 4; ж) –5; з) 0; и) 0; к) 2; л) ошибка; м) ошибка

1.6. а) 3; б) 3; в) 9; г) 9; д) 4; е) –2; ж) 2; з) –4; и) 3; к) –3; л) 0; м) 3

1.9. Неправильные: а), г) (это целое число), д), е)

1.10. Да

1.13. а) 5.0; б) 2.0; в) 6.38; г) –0.7444; д) 2.75; е) –0.1667; ж) 1.4142; з) 3.1416; и) 5E6; к) –24.8E–7; л) 1E6; м) 1E–5

1.14. а) –2.7; б) 0.666; в) 10.0; г) 0.1

1.15. Неправильные: в), г), е), ж), з), л), м)

1.17. Целые: а) 1; и) 2; л) 0; Вещественные: б) 1.0; в) –9.0; г) 0.0; д) 2.0; е) 2.0; ж) 2.0; з) 2.0; Ошибки: к), м)

1.18. а) –2; б) –2.0; в) 25; г) 25.0; д) 4.0; е) 4.0; ж) 1.0; з) 1.0

1.19. а) 6; б) 7; в) 6; г) 6; д) –1; е) –2; ж) 2; з) –2; и) 0; к) 1; л) 0; м) –1

1.25. а) a*b div c mod (a+b); б) (–n) mod m;
в) sin(x+y)/2–cos((x+y)/2); г) a*x/(b*y)+a/x/(b/y)

1.26. а) 32.0; в) –1

1.27. 7 операций; (x+0.5)*(y+0.7)–0.75

1.28. а) a+b*x+c*y*z; б) ((a*x–b)*x+c)*x–d; в) a*b/c+c/(a*b);
г) (x+y)/a1*a2/(x–y); д) (1+x/2+y/6)/(1+2/(3+x*y)); е) abs(a+b*x);
ж) 1E4*alpha–3.2*beta; з) sqr(1+x); и) sqrt(1+sqr(x));
к) ln(exp(x)+exp(–x)); л) sin(8); м) sqr(cos(x*x*x))

1.29. а) $\frac{p+q}{r+s}-\frac{pq}{rs}$; б) $10^3+\frac{\beta}{x_2-\gamma\delta}$

1.30. б) ln(x/5)/ln(2); д) arctan(x/sqrt(1–x*x)); е) 1/x; ж) sqr(sqr(x));
к) exp(100*ln(x)); з) exp(ln(1+x)/3); п) ((x+y)+abs(x–y))/2

1.31. exp(1); 4*arctan(1)

1.32. sin(3.1415927*x/180)

1.34. а) 1; б) 432; в) 2; г) 2

1.37. 12 (N и n – одно и то же имя)

1.38. Правильно: б)

1.40. а) y:=1+x*(1+x/2*(1+x/3*(1+x/4)))

1.41. в) d:=sqrt(sqr(x1–x2)+sqr(y1–y2));
д) p:=(a+b+c)/2; d:=sqrt(p*(p–a)*(p–b)*(p–c));
е) d:=x–trunc(x)

1.44. Правильные: а), г), д)

1.45. а) Неправильно: если k – вещественная переменная, тогда недопустима операция k *mod* 3, а если k – целая переменная, тогда ей присваивается значение вещественного выражения ($cos(0)$ – вещественное число), что также недопустимо.

1.46. h:=k div 100 mod 10 или h:=k mod 1000 div 100

2. ЛОГИЧЕСКИЙ ТИП

2.2. а) true; б) true; в) false; г) true; д) false

2.3. а) k mod 7=0; б) sqr(b)–4*a*c<0; в) sqr(x–1)+sqr(y)>sqr(r);
г) abs(x)<=1; д) a<>b; е) sqr(round(sqrt(n)))=n

2.4. Да.

2.5. а) true; б) false; в) false; г) true; д) true; е) ошибка;
ж) ошибка; з) true; и) 0; к) 1; л) true; м) true;
н) false; о) –1; п) ошибка; р) ошибка

2.6. а) false; б) true; в) false; г) true; д) false; е) false; ж) true; з) true; и) false

2.7. а) a; б) not a; в) a

2.9. в) обязательно; г) все ошибочны

2.10. а) (x>0) and (x<1); б) (x>=y) and (x>=z); г) a or b; д) a and b

2.11. в) (a and b) or ((not c) and a); г) (a and b) > (a or b)

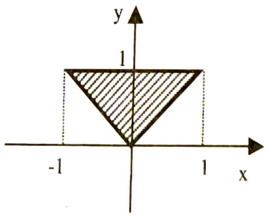


Рис. 24

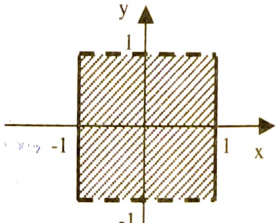


Рис. 25

2.12. в) (x>=2) and (x<=5) or (abs (x)<=1);
г) (x<–1) or (x>1) and (x<2) or (x>5);
к) (y mod 400 = 0) or (y mod 4 = 0) and (y mod 100 <> 0)

2.13. Верно. Ошибка в обоих выражениях.

2.14. а) см. рис. 24; е) см. рис. 25

2.15. в) (sqr(x)+sqr(y)<=1) and (abs(x)+abs(y)>=1)

2.16. д)

a	b	a<=b	not a	not a or b
true	true	true	false	true
true	false	false	false	false
false	true	true	true	true
false	false	true	true	true

2.17. а) not a and b

3. ВВОД-ВЫВОД. ПРОСТЕЙШИЕ ПРОГРАММЫ

3.2. а) и б) k=–25, x= 3.14; в) x=5.0, k=6; г) ошибка;
д) k=4; е) и ж) ошибка

3.4. а) 7; б) false; в) k+1; г) k=3; д) 123; е) 12 3; ж) truefalse;
з) true false

3.5. write('>'); read(x)

3.7. (Пробелы явно обозначены знаком _)
а) _ _ 123; б) 123; в) 123; г) _ _ 124; д) 1234; е) _ 1234;
ж) 123 _ _ _ 4; з) _ 123 _ _ _ 4

3.9. write('*', '*':11, '*':18)

3.10. (Пробелы явно обозначены знаком _)
а) _ 1.235E+01; б) –1.235E+01; в) _ 1.2345600E+01;
г) –1.2345600E+01; д) _ 1.2E+01; е) –1.2E+01

3.12. (Пробелы явно обозначены знаком _)
а) _ _ _ _ 12.346; б) _ _ _ –12.346; в) _ 12.345600;
г) –12.345600; д) _ _ _ _ _ _ 12.346; е) _ _ _ _ _ –12.346;
ж) 12.346; з) –12.346; и) 12.3

3.13. в) 1 23
4 5
6 7

3.16. x1= –1.00 x2= –2.00

3.17.
```
program коэфф (input, output);
  var x1, x2, p, q: real;
begin
  read(x1, x2);   {ввод корней}
  p:= –(x1+x2); q:=x1*x2;  {вычисление коэффициентов}
  writeln('коэффициенты:  1.0', p:7:3, q:7:3)
end.
```

3.22. Ошибки:
а) в n=n+1 справа от = указано выражение;
б) в s=pi/2 справа от = указано выражение;
в) имя *star* используется до описания.

3.23. Достаточно в разделе констант число 2.71828 заменить на число 3.14159.

3.24.
```
program больше (output);
  const e=2.71828; pi=3.14159;
begin writeln(pi>e*ln(pi)) end.
```

3.26. а) В левой части оператора присваивания (*d:=sqr(d)*) нельзя указывать имя константы.

б) Применять функцию *ord* к вещественному аргументу (*x*) нельзя. В выражении *ord(x)=k* число сравнивается с логической величиной, что недопустимо.

в) Переменная *c* не получила значения, поэтому прибавлять к ней ничего нельзя.

г) Переменная *y* не описана.

д) Имя *B* описано дважды – как константа и как переменная (*B* и *b* – это одно и то же имя).

4. ОПЕРАТОРЫ: ПУСТОЙ, СОСТАВНОЙ, УСЛОВНЫЙ, ПЕРЕХОДА

4.2. Пустой оператор есть во всех примерах: а) между *else* и «;»; б) между *then* и *else*; в) между «;» и *end*; г) между первой и второй точками с запятой; д) между *begin* и «;», между «;» и *end*; е) между *begin* и *end*

4.3. а1) x=1, y=4; а2) x= –4, y=4; б1) x=1, y=4; б2) x= –4, y=2;
в1) x=1, y=2; в2) x= –4, y=2; г1) x=1, y=2; г2) x= –4, y=4;
д1) x=1, y=4; д2) x=3, y=2

4.4. а) 1, 2, 1, 3; д) 1, 3, 2, 3; е) 1, –4, 2, –4

4.5.
```
а) if (x>0) and (x<2) then y:=sqr(cos(x)) else y:=1–sin(sqr(x));
б) if abs(a)< =3.14159/2 then x:=exp(sin(a)–1);
г) if (a>b) and (a>c) then d:=a else if b>c then d:=b else d:=c;
д) if x<0 then if x>y then z:=x else z:=y
        else if x<y then z:=x else z:=y
```

4.7.
```
а) c:=y div 100; if y mod 100 <> 0 then c:=c+1;
в) d:=sqr(b)–4*a*c; t:=d>=0;
   if t then
     begin d:=sqrt(d); a2:=2*a; x1:=(–b+d)/a2; x2:=(–b–d)/a2 end;
д) if a<b then begin r:=a; a:=b; b:=r end;    {a>=b}
   if a<c then begin r:=a; a:=c; c:=r end;    {a>=c}
   if b<c then begin r:=b; b:=c; c:=r end;    {b>=c}
```

4.8. Оба варианта верные, но второй короче.

4.10.
```
if a then x:=true else  if b then x:=c else x:=false
```

4.14. 6

4.15.
```
а)  1-й вариант:  { при a>b НОД(a,b)=НОД(a–b,b) }
    a1:=a; b1:=b; {чтобы не портить a и b}
1:  if a1=b1 then goto 2;
    if a1>b1 then a1:=a1–b1 else b1:=b1–a1;
    goto 1;
2:  c:=a1;
    2-й вариант:
      {при a>b>0 НОД(a,b)=НОД(a mod b,b); НОД(0,b)=b }
    a1:=a;  b1:=b;
1:  if (a1=0) or (b1=0) then goto 2;
    if a1>b1 then a1:=a1 mod b1 else b1:=b1 mod a1;
    goto 1;
2:  if a1=0 then c:=b1 else c:=a1;

б)  n:=0;
1:  n:=n+1;
    u:=cos(cos(n)/sin(n));  if u>=0 then goto 1;

в)  p:=1;  i:=2;
1:  p:=p*(1–1/sqr(i));
    i:=i+1;
    if i<=n then goto 1;

г)  n:=40;  y:=cos(n);
1:  n:=n–1;  y:=cos(n+y); if n>1 then goto 1
```

4.16.
```
program max (input, output);
  label 4, 9;
  const n=50; {количество заданных чисел}
  var x, max: real;  {x – очередное число, i – его номер,}
      i: integer; {max – наибольшее среди уже введенных чисел}
begin
  read(max);  i:=1;
4: i:=i+1;  if i>n then goto 9;
  read(x);  if x>max then max:=x;
  goto 4;
9: writeln('max=', max:1:5)
end.
```

4.21.
```
program geom (input, output);
   label 1, 99;
   const n=20;
   var k: integer;  p, x: real;
begin
   k:=0;  p:=1;
1: read(x);  k:=k+1;  {очередное число и его номер}
   if x<=0 then  begin writeln('ошибка: число <= 0'); goto 99 end;
   p:=p*x;  if k<n then goto 1;
   writeln(exp(ln(p)/n):1:5);
99:
end.
```

5. ОПЕРАТОР ЦИКЛА

5.1. г) Значение k: 1) 5; 2) 1; 3) 4

5.2. в) Значение k: 1) 0; 2) 1; 3) 4

5.3. д) 3 раза;
з) значение m: 1) 1; 2) 2; 3) не определено; 4) 3

5.4.
```
{оператор цикла с предусловием:}
f:=1; i:=2; while i<=10 do begin f:=f*i; i:=i+1 end;
{оператор цикла с постусловием:}
f:=1; i:=2; repeat f:=f*i; i:=i+1 until i>10;
{операторы цикла с параметром:}
f:=1; for i:=2 to 10 do f:=f*i;
f:=1; for i:=10 downto 2 do f:=f*i
```

5.5. а)
```
a1:=a; b1:=b; {чтобы не портить a и b}
while a1<>b1 do if a1>b1 then a1:=a1–b1 else b1:=b1–a1;
c:=a1;
```
б) `n:=0; repeat n:=n+1; u:=cos(cos(n)/sin(n)) until u<0;`
в) `p:=1; for i:=2 to n do p:=p*(1–1/sqr(i));`
г) `y:=cos(40); for n:=39 downto 1 do y:=cos(n+y)`

5.6. б) `s:=1; k:=1; for j:=2 to n do begin k:= –k; s:=s+k/j end;`
в) `p:=1; while p<x do p:=10*p;`
д) `s:=0; m:=n; repeat s:=s + m mod 10; m:=m div 10 until m=0`

5.7. б)
```
program difference (input, output);
  const n=100;
  var max, min, x: real; j: integer;
begin
read(max); min:=max; {начальное значение для max и min}
  for j:=2 to n do
    begin
      read(x);
      if x>max then max:=x; if x<min then min:=x
  end;
  writeln(max–min:1:5)
end.
```
в)
```
program MinIndex (input, output);
  var min, n, k, i: integer;
begin
  read(min); n:=1; {начальное значение min и его номер}
  read(k); i:=2; {очередное число и его номер}
  while k<>0 do
    begin if k<min then begin min:=k; n:=i end; read(k); i:=i+1 end;
  writeln(n)
end.
```

5.8. а) `y:=1; for i:=2 to 11 do y:=y*x+i;`

б) `y:=11; for i:=10 downto 1 do y:=y*x+i`

5.11. а) `y:=1; for i:=1 to n do y:=2*i*y;`
е) `y:=0; for i:=0 to 10 do y:=y+sin(1+0.1*i)`

5.15. `s:=0; for k:=trunc(ln(x))+1 to trunc(exp(x)) do s:=s+sqr(k)`

5.19. а)
```
y:=0; u:=1; {u = x^i =x*x^(i–1)}
for i:=1 to 30 do begin u:=x*u; y:=y+cos(u) end;
```
б)
```
y:=1; f:=1; {f = i! =(i–1)!*i}
for i:=2 to n do begin f:=f*i; y:=y+f end
```

5.20. а)
```
y:=1; {сумма} n:=0; {номер слагаемого}
u:=1; {n-е слагаемое = x^n/n! = x^(n-1)/(n–1)!*(x/n)}
repeat n:=n+1; u:=u*x/n; y:=y+u until abs(u)<eps
```

5.21. а)
```
g:=1; f:=1; {g=f0, f=f1}
for n:=2 to 40 do
  begin {g=fn-2, f=fn-1} h:=g; g:=f; f:=h+g {g=fn-1, f=fn} end
```

5.22.
```
{поиск корня x на отрезке [a, b]=[π, 3π/2–δ]: a–tg(a)>0, b–tg(b)<0}
  a:=3.1415926; b:=4.71;
  repeat
    c:=(a+b)/2; {середина отрезка [a, b]}
    if c–sin(c)/cos(c)>0 then a:=c { [a, b]:=[c, b] }
                         else b:=c { [a, b]:=[c, b] }
  until b–a<eps;
  x:=(a+b)/2 { корень – середина последнего отрезка [a, b] }
```

5.25. а)
```
k:=1;
while (k<=10) and (x<=y) do begin x:=2*x; y:=y/2; k:=k+1 end;
```
б)
```
k:=1; exit:=false;
while (k<=10) and not exit do
  begin x:=2*x; y:=y/2; if x>y then exit:=true else k:=k+1 end
```

5.26. а)
```
p:=true;
for k:=2 to trunc(sqrt(n)) do
  if n mod k = 0 then begin p:=false; goto 1 end;
1:
```
б) `p:=true; k:=2; while p and (sqr(k)<=n) do if n mod k = 0 then p:=false else k:=k+1`

5.30. а)
```
program increasing (input, output);
  label 99;
  const n=25;
  var a, b, j: integer; {a – предыдущее число, b – очередное}
begin
  read(b);
  for j:=2 to n do
    begin
      a:=b; read(b); if a>=b then begin writeln('Нет'); goto 99
      end;
```

```
    writeln('Да');
  99:
  end.
```

5.31. Ошибка в решении а): в переменной *s* накапливается сумма не очередной десятки чисел, а всех предыдущих чисел; значение *s* надо обнулять в начале каждой десятки.

5.34.
```
max := -1;
for i:=1 to 30 do
 for j:=i to 30 do
  begin m:=cos(i*i+j*j); if m>max then max:=m end
```

5.36.
```
k:=0;
{d1 – левая, d2 – средняя, d3 – правая цифры числа}
for d1:=1 to 9 do
  for d2:=0 to 9 do
    for d3:=0 to 9 do
      if d1+d2+d3=n then k:=k+1
```

6. СИМВОЛЬНЫЙ ТИП

6.6. а) true; б) 5; в) ошибка; г) true; д) true; е) false; ж) true; з) 23; и) '9'; к) ошибка

6.7. а) t:=d='*'; б) t:=(d='a') or (d='q'); в) t:=(d>='0') and (d<='9')

6.8. а) '+'; б) 'c'

6.9.
```
s:=ord('S')+ord('U')+ord('M')
```

6.10.
```
write(chr(65), chr(71), chr(69))
```

6.11.
```
if dig='9' then next:='0' else next:=succ(dig)
```

6.12.
```
b:=ord('z')-ord('a')=25
```

6.13.
```
writeln; for c:='A' to 'Z' do write(c); writeln
```

6.14. а)
```
for d:='1' to '9' do
    begin
     for c: ='1'  to pred(d) do write('0'); write(d);
     for c:=succ(d) to '9' do write('0');   writeln
    end
```

6.15.
```
program ab (input, output);
 var c: char; {очередной символ текста}
     ka, kb: integer; {число вхождений a и b}
begin
 ka:=0;  kb:=0; read(c);
 while c<>'.' do
  begin if c='a' then ka:=ka+1 else if c='b' then kb:=kb+1; read(c) end;
 writeln(ka>kb)      end.
```


6.18.
```
program ЦелоеЧисло (input, output);
 var c: char;  k: integer;
begin
 read(c);  if (c='+') or (c='–') then read(c);  {пропуск знака}
 k:=0;
 while (c>='0') and (c<='9') do begin k:=k+1; read(c) end;
  {подсчет количества цифр}
 if (k>0) and (c='.') then writeln ('правильно') else writeln ('неправильно')
end.
```

6.20.
```
program th (input, output);
 label 99;
 var k: integer;  a, b: char;  {a – очередной символ текста, b – следующий}
begin
 k:=0;
 read(a);  if a='.' then goto 99;  {пустой текст}
 read(b);
 while b<>'.' do
  begin if (a='t') and (b='h') then k:=k+1; b:=a;  read(b) end;
99:  writeln('Число вхождений th: ', k)
end.
```

6.23. а)
```
program БезЦифр (input, output);
  var c: char;
 begin
  read(c);
  repeat
    if (c='+') or (c='–') then write(c,c) else
    if (c<'0') or (c>'9') then write(c);
    read(c)
  until c='.';
  writeln
 end.
```

б)
```
program БезПлюса (input, output);
  var a, b: char; {a – очередной символ, b – следующий}
 begin
  read(a,b);
  while b<>'.' do
   begin {вывод, если надо, только a}
    if (a<>'+') or (b<'0') or (b>'9') then write(a); a:=b;  read(b)
  end;
  writeln(a)
 end.
```

6.26. а)
```
program БуквaA (input, output);
  var c: char; k: integer;
 begin
  k:=0;
```

```
repeat {цикл по словам}
 read(c); {первая буква слова}
 if c='a' then k:=k+1; {анализ первой буквы}
 repeat read(c) until (c=',') or (c='.') {пропуск остальных}
until c='.'; {признак последнего слова}
writeln ('С буквы "a" начинается ', k, ' слов')
end.
```

6.27. {Ответ на вопрос: d:=ord(c)–ord('0')}
n0:=ord('0'); k:=100*(ord(c2)–n0)+10*(ord(c1)–n0)+(ord(c0)–n0)

6.28. {Ответ на вопрос: c:=chr(ord('0')+d)}
```
n0:=ord('0');
d:=k div 100; c2:=chr(n0+d);
d:=k mod 100 div 10; c1:=chr(n0+d);
d:=k mod 10; c0:=chr(n0+d)
```

6.29.
```
{var c: char; sign, n0: integer;}
 {определение знака числа:}
sign:=1; read(c);
if c='–' then begin sign:= –1; read(c) end else if c='+' then read(c);
{ввод цифр (первая – в c) и вычисление модуля числа по схеме Горнера:}
n0:=ord('0'); k:=0; repeat k:=10*k+ord(c)–n0; read(c) until c=' ';
k:=sign*k {учет знака}
```

6.38. а)
```
writeln; writeln('*': k);
```
б)
```
writeln;
 if k<n then writeln('*': k, 'T': n–k) else
  if k=n then writeln('*': k) else writeln('T': n, '*': k–n)
```

7. ПЕРЕЧИСЛИМЫЕ И ОГРАНИЧЕННЫЕ ТИПЫ. ОПЕРАТОР ВАРИАНТА

7.3. а) Допустимые присваивания: 1), 2) и 3);
б) 1) true; 2) true; 3) false; 4) ошибка; 5) лето; 6) зима; 7) ошибка; 8) тепло; 9) 0; 10) 3; 11) ошибка; 12) –1
в) допустим;
г) допустимо только 3).

7.4. Правильно описаны только типы *гласная*, *фигура* и *лог*.

7.5. t:=(m1<m2) or (m1=m2) and (d1<d2)

7.7. а) if m=дек then m1:=янв else m1:=succ(m);
в) m1:=m;
```
 for i:=1 to k mod 12 do if m1=дек then m1:=янв else m1:=succ(m1);
```

7.8. m:=пики; for i:=1 to n do m:=succ(m)

7.9. cap:=Вена; for i:=1 to ord(st) do cap:=succ(cap)

7.13. а)
```
if x=черный then write('черный') else
 if x=серый then write('серый') else write('белый');
```


б)
```
{var ch: char;}
read(ch); if ch='a' then y:=a else if ch='b' then y:=b else y:=c
```

7.15. а) Допустим только оператор 1);
б) допустим;
в) 1) всегда допустимо; 2) и 3) допустимы, если значения *n* и *k*–1 принадлежат диапазону [0,9];
г) будет ошибка.

7.16. Правильно описаны типы *цифра*, *угол*, *плюс*, *неделя* и *будни*.

7.18. а) p=false, d=3; б) p=true, d=235; в) p=true, d=1; г) ошибка

7.20.
```
case m of
 янв, фев, дек:  s:=зима;
 мар, апр, май:  s:=весна:
 июн, июл, авг: s:=лето;
 сен, окт, ноя:  s:=осень
end
```

7.28.
```
case m of
 янв, мар, май, июл, авг, окт, дек: d:=31;
 фев: d:=28;
 апр, июн, сен, ноя: d:=30
end
```

7.31. а)
```
{var m1: месяц;}
{k – количество дней в месяцах, предшествующих месяцу m:}
  k:=0; m1:=янв;
  while m1<m do
   begin
     case m of
       янв, мар, май, июл, авг, окт: k:=k+31;
       фев: k:=k+29;
       апр, июн, сен, ноя: k:=k+30
     end;
     m1:=succ(m1)
   end;
  k:=k+d {учет дней в месяце m}
```

8. РЕГУЛЯРНЫЕ ТИПЫ: ВЕКТОРЫ

8.1. г) Нельзя использовать б) и 8).

8.2. Массив *a*: а) 30; б) вещественные числа; в) 1 и 30;
Массив *b*: а) 5; б) *x*, *y*, *z*; в) –2 и 2;
Массив *c*: а) 10; б) массивы типа *вектор*; в) '0' и '9';
Массив *d*: а) 3; б) целые от 0 до 23; в) *вчера* и *завтра*.

8.3.
```
type R = array [char] of 1..maxint;
```

8.4. Неправильно описаны типы *слово*, *вектор* и *цифры*, а также переменная *x* (везде недопустимый тип индекса).

8.5. в) Правильные: 1) и 5).
е) Правильные: 1), 3) и 5).

8.6.
```
а) y:=1; for i:=1 to n do y:=y*x[i]; y:=exp(ln(abs(y))/n);
б) y:=x[1]; for i:=2 to n do if x[i]>y then y:=x[i];
в) y:=0; p:=1; for i:=1 to n do begin y:=y+p*x[i]; p:= –p end;
г) y:=0; for i:=1 to n do y:=y+x[i]*x[n+1–i];
д) y:=0; for i:=0 to (n–1) div 2 do y:=y+sqr(x[2*i+1]);
е) y:=1; p:=0; {предыдущий сомножитель}
   for i:=n downto 1 do begin p:=p+x[i]; y:=y*p end;
```

8.7.
```
n:=101; for k:=102 to 118 do if av[k]>av[n] then n:=k
```

8.11.
```
for i:=1 to 72 do t[i]:=k[t[i]]
```

8.12.
```
{var wd: ДеньНедели; j: integer;}
wd:=ср; {очередной день недели}
for j:=1 to 365 do
 begin год[j]:=wd; if wd=вс then wd:=пн else wd:=succ(wd) end
```

8.13.
```
а) {var max: integer; и1: имя;}
   max:=139;
   for и1:=Валя to Шура do
    if (Пол[и1]=муж) and (Рост[и1]>max) then
     begin max:=Рост[и1]; И:=и1 end
```

8.16.
```
а) for i:=1 to n do read(x[i]);
д) eq:=true; for i:=1 to n do  if x[i]<>y[i] then begin eq:=false; goto 1 end;
   1:
е) y:=x
```

8.19. Ошибка в варианте *б*: если *p* нет в массиве *x*, то в заголовке *while*-цикла с *p* будет сравниваться несуществующий элемент *x*[11].

8.21. Массивы не нужны в задачах а) и г).

8.25.
```
а) program diff (input, output);
     var c: char;  k: integer;
         d: array['0'..'9'] of boolean; {d[c]=true, если цифра c входит в текст}
   begin
    for c:='0' to '9' do d[c]:=false;
    read(c); while c<>'.' do begin d[c]:=true; read(c) end;
    k:=0; for c:='0' to '9' do if d[c] then k:=k+1;
    writeln(k)
   end.
```

8.26.
```
program reverse (input, output);
 const n=70;
 var c: char;  k: integer; t: array[1..n] of char;
begin
 k:=0;  {число введенных символов}
 read(c);
 while c<>'.' do begin k:=k+1; t[k]:=c; read(c) end;
```



```
 for k:=k downto 1 do write(t[k]); writeln
end.
```

8.29.
```
б) for k:=1 to n div 2 do  begin r=x[k]; x[k]:=x[n+1–k]; x[n+1–k]:=r end;
г) r:=x[1];  for k:=1 to n–1 do x[k]:=x[k+1];  x[n]:=r
```

8.31.
```
а)  k:=0;
     for i:=1 to n do
      begin p:=1; while p<s[i] do p:=2*p; if p=s[i] then k:=k+1 end
```

8.34.
```
 {label 9;  var ф1, ф2: фамилия;}
ЕстьТезки:=true;
for ф1:=Бетелин to pred (Школьный) do
 for ф2:=succ(ф1) to Школьный do
  if MM511[ф1]=MM511[ф2] then goto 9;
ЕстьТезки:=false;
9:
```

8.37.
```
p:=1; {индекс очередного элемента из z} i:=1; {из x}  j:=1; {из y}
 {пока есть нерассмотренные элементы и в x, и в y:}
repeat
  if x[i]<y[j] then begin z[p]:=x[i]; i:=i+1 end
               else begin z[p]:=y[j]; j:=j+1 end;
  p:=p+1
until (i>k) or (j>m);
{x или y исчерпан → перепись в z оставшихся элементов другого массива:}
if i>k then repeat z[p]:=y[j]; j:=j+1; p:=p+1 until j>m
       else repeat z[p]:=x[i]; i:=i+1; p:=p+1 until i>n
```

8.38. 1-й вариант: а) 2*5=10 сравнений; б) это индекс *a*;
2-й вариант: а) 5+1=6 сравнений; б) это индекс *a*+1, если *p*>*x*[*n*], или индекс *a* иначе

8.41.
```
а)  for k:=n downto 2 do
      begin
        {поиск m – номера максимума в x[1..k]:}
       m:=1;  for i:=2 to k do if x[i]>x[m] then m:=i;
        {перестановка элементов x[m] и x[k]:}
       r:=x[k];  x[k]:=x[m]; x[m]:=r
      end
```

8.50.
```
а) R[0]:=P[0]; for i:=1 to n do R[i]:=P[i]–a*P[i–1]; R[n+1]:= –a*P[n]
```

9. РЕГУЛЯРНЫЕ ТИПЫ: МАТРИЦЫ

9.2.
```
program ВводВыводМатрицы (input, output);
  const n=4;
  var A: array [1..n, 1..n] of real;  i, j: integer;
begin
{ввод матрицы по строкам:}
  for i:=1 to n do for j:=1 to n do read(A[i,j]);
```

```
{вывод матрицы по столбцам:}
 for j:=1 to n do begin for i:=1 to n do write(A[i,j]:10:5); writeln end
end.
```

9.3. а) `for i:=1 to n do for j:=1 to n do C[i,j]:=A[i,j]+B[i,j];`

б)
```
{ y[i]=∑A[i,j]*x[j] }
 for i:=1 to n do
  begin s:=0; for j:=1 to n do s:=s+A[i,j]*x[j]; y[i]:=s end;
```

в)
```
{ C[i,j]=∑A[i,k]*B[k,j] }
 for i:=1 to n do
  for j:=1 to n do
   begin
    s:=0; for k:=1 to n do s:=s+A[i,k]*B[k,j]; C[i,j]:=s
   end;
```

г)
```
{B[i,j] <-> B[j,i], j>i}
 for i:=1 to n–1 do
  for j:=i+1 to n do
   begin r:=B[i,j]; B[i,j]:=B[j,i]; B[j,i]:=r end
```

9.4.
```
t:=false;
for i:=0 to 9 do for j:= –5 to 3 do if A[i,j]<>B[i,j] then goto 1;
t:=true;
1:
```

9.5. б)
```
m:=B[2,1];
 for i:=3 to n do for j:=1 to i–1 do if B[i,j]>m then m:=B[i,j]
```

в)
```
r:=1; c:=1; m:=B[1,1];
 for i:=1 to n do
  for j:=1 to n do
   if B[i,j]>m then begin m:=B[i,j]; r:=i; c:=j end;
```

9.6.
```
d:=0;
for i:=1 to 39 do
 for j:=i+1 to 40 do
  begin
   r:=sqr(M[i,x]–M[j,x])+sqr(M[i,y]–M[j,y]); if r>d then d:=r
  end;
d:=sqrt(d)
```

9.9. Допустимы: а), в).

9.10. а)
```
{типы строк матрицы A и вектора x одинаковы}
 for i:=0 to 9 do A[2*i+1]:=x;
```

б) `for j:=1 to 10 do for i:=1 to 20 do A[i,2*j]:=x[i];`

в)
```
{типы строк матрицы B и вектора x различны}
 for i:=1 to 6 do for j:=1 to 20 do B[i,j]:=x[j]
```

9.13. а)
```
program sort (input, output);
 const n=20; m=6;
 type строка = array[1..m] of real;
      матрица = array [1..n] of строка;
```



```
 var A: матрица; x: строка; i, j, k: integer;
begin
  {ввод A:}
 for i:=1 to n do for j:=1 to m do read(A[i,j]);
  {сортировка выбором:}
 for k:=n downto 2 do
  begin
   {поиск j – номера max A[1..k,1]:}
   j:=1; for i:=2 to k do if A[i,1]>A[j,1] then j:=i;
   {перестановка k-й и j-й строк:}
   x:=A[k]; A[k]:=A[j]; A[j]:=x
  end;
   {вывод A:}
 for i:=1 to n do
  begin for j:=1 to m do write(A[i,j]:10:5); writeln end
end.
```

9.14. а)
```
for i:=1 to 10 do A[1,i]:=0;
for i:=2 to 10 do begin A[i]:=A[1]; A[i,i]:=i–1 end
```

9.17. а)
```
s:=A[1,1]+A[9,1]+A[1,9]+A[9,9];
for i:=2 to 8 do s:=s+A[1,i]+A[9,i]+A[i,1]+A[i,9]
```

9.18. а)
```
for k:=1 to 15 do
 begin
  j:=0; repeat j:=j+1 until (A[k,j]<>0) or (j=20); b[k]:=A[k,j]=0
 end
```

9.24. а)
```
{label 1; var P, ДР: имя;}
  {поиск P – ребенка И, а затем поиск ДР – дочери P}
 for P:=Анна to Юрий do
  if (ТР[И,P]=сын) or (ТР[И,P]=дочь) then
   for ДР:=Анна to Юрий do
    if ТР[P,ДР]=дочь then begin B:=ДР; goto 1 end;
1:
```

10. РЕГУЛЯРНЫЕ ТИПЫ: СТРОКИ

10.1. Строковым является только тип *b*.

10.2. в) Правильно: 1); д) Правильно: 1) и 2).
з) 1) false, 2) true, 3) и 4) – ошибки.

10.3.
```
program ВводВыводСтроки (input, output);
 const n=60;
 var s: packed array [1..n] of char; i: integer;
begin
 for i:=1 to n do read(s[i]);
 writeln(s); writeln(s)
end.
```

10.4.
```
if a>=b then begin c:=a; a:=b; b:=c end
```

10.5. а) for i:=1 to 60 do if C[i]<>'hello' then write(C[i],' ');

в) for i:=1 to 60 do write(C[i,5])

10.6.
```
 {обработка массива a1:}
k1:=0;
for i:=2 to 20 do
 begin
  j:=0; repeat j:=j+1 until (a1[i,j]<>a1[1,j]) or (j=8);
  if a1[i,j]=a1[1,j] then k1:=k1+1
 end;
 {обработка массива a2:}
k2:=0;
for i:=2 to 20 do if a2[i]=a2[1] then k2:=k2+1
```

10.7.
```
{var c: char;  i: integer;}
{сначала запись 6 пробелов в s, а затем замена первых
 пробелов введенными буквами:}
s:='      '; {6 пробелов}
i:=0; read(c); while c<>'.' do begin i:=i+1; s[i]:=c; read(c) end
```

10.9.
```
{type string = packed array [1..6] of char;
var n: array [0..5] of string;
    y: string;  i: integer;  c: char;  ok: boolean;}
 {формирование массива из названий цвета в виде строк:}
n[0]:='red   ';  n[1]:='blue  '; n[2]:='green ';
n[3]:='yellow'; n[4]:='black '; n[5]:='white ';
 {ввод не более 6 букв до пробела и запись их в y:}
y:='      '; i:=0; read(c);
while (c<>' ') and (i<6) do begin i:=i+1; y[i]:=c; read(c) end;
ok:=(i>0) and (c=' '); {от 1 до 6 букв и пробел за ними?}
if ok then
 begin {если y=n[i], то x:=i-е имя типа color:}
   i:=-1; repeat i:=i+1 until (y=n[i]) or (i=5);
   if y<>n[i] then ok:=false else
    begin x:=red;  for i:=0 to i-1 do x:=succ(x) end
 end;
if not ok then writeln ('неверное значение типа color')
```

10.10.
```
{var v1: packed array [1..5] of char;  i, j: integer;}
k:=0;  v1:=v; {копирование v в v1,  т.к. v[j] – ошибка}
for i:=1 to 200 do
 begin
  j:=0; repeat j:=j+1 until (s[i]=v1[j]) or (j=5); if s[i]=v1[j] then k:=k+1
 end
```

10.12. а)
```
program РусскиеБуквы (input, output);
   const n=60;
   var rus: packed array [1..33] of char;  c: char;  i, j: integer;
```



```
begin
 rus:='абвгдеёжзийклмнопрстуфхцчшщъыьэюя';
 for i:=1 to n do
  begin
   read(c);
    {поиск c в rus:}
   j:=0;  repeat j:=j+1 until (c=rus[j]) or (j=33);
   if c=rus[j] then write(c)
  end;
 writeln
end.
```

10.15. б)
```
{поиск в s буквы 'w' или пробела:}
i:=1; while (s[i]<>'w') and (s[i]<>' ') do i:=i+1;
if s[i]='w' then
   {сдвиг следующих за 'w' букв на 1 позицию влево и
    запись пробела вместо последней буквы: }
   repeat i:=i+1; s[i-1]:=s[i] until s[i]=' ';
```

г)
```
{поиск в s буквы 'x' или пробела:}
i:=1; while (s[i]<>'x') and (s[i]<>' ') do i:=i+1;
if s[i]='x' then
 begin
  s[i]:='k'; {замена 'x' на 'k'}
  {вставка 's' за 'k' и сдвиг последующих букв вправо:}
  b:='s'; {вставляемый символ}
  repeat
   i:=i+1; {куда вставлять}
   c:=s[i]; {освободить место для b}
   s[i]:=b; {вставка b}
   b:=c {новый вставляемый символ}
  until b=' '
 end
```

11. ПРОЦЕДУРЫ И ФУНКЦИИ

11.2. а)
```
procedure сокр (a, b: integer; var p, q: integer);  {a, b > 0}
  var a1, b1: integer;
begin
  a1:=a; b1:=b;
  while a1<>b1 do if a1>b1 then a1:=a1-b1 else b1:=b1-a1;
  p:=a div a1;  q:=b div a1
end;
  . . .
c:=0;  d:=1;  {c/d=0}
for i:=1 to 20 do сокр(c*i+d, d*i, c, d)  {c/d + 1/i = (c*i+d)/(d*i)}
```

в)
```
procedure maxmin (var x, y: real);
  var r: real;
```

```
begin if x<y then begin r:=x; x:=y; y:=r end end;
  ...
maxmin(a,b); maxmin(a,c); {a=max}
maxmin(b,c); {c=min}
```

в)
```
procedure minus13;
  const n=13;
  var k: integer;
begin for k:=1 to n do write('–'); writeln end;
  ...
minus13; writeln('  a   not a'); minus13;
for a:=true downto false do writeln(a:6, not a:6);
minus13;
```

11.3.
```
program площадь (input, output);
  var a, b, c, d: real;
  procedure печплощ (x, y, z: real);
    var p: real;
  begin
    if (x+y>z) and (y+z>x) and (z+x>y) then
      {из этих неравенств следует: x, y, z > 0}
    begin p:=(x+y+z)/2; writeln(sqrt(p*(p–x)*(p–y)*(p–z)):10:5) end
  end;
begin
  read(a,b,c,d);
  печплощ(a,b,c); печплощ(a,b,d); печплощ(a,c,d); печплощ(b,c,d)
end.
```

11.6. б) 2 3 2

11.7. б) 2 3 3
в) 0 2

11.10.
```
procedure sum (var x, y, z: вектор);
  var i: integer;
begin for i:=1 to n do z[i]:=x[i]+y[i] end;
  ...
sum(a,b,d); sum(d,c,d);
```

11.12. в) f(1)=1, f(2)=2, f(3) – значение не определено при этом аргументе

11.13. б)
```
function sign (a: real): integer;
  begin
    if a<0 then sign:= –1 else if a>0 then sign:=1 else sign:=0
  end;
  ...
z:=(sign(x)+sign(y))*sign(x+y)
```

11.15.
```
function скал (var x, y: вектор): real;
  var s: real; i: integer;
begin s:=0; for i:=1 to n do s:=s+x[i]*y[i]; скал:=s end;
```



```
  ...
t:=(скал(a,b)=0) and (скал(a,c)=0) and (скал(b,c)=0)
```

11.17.
```
program shtg (input, output);
  var x, y: real;
  function sh (z: real): real; begin z:=exp(z); sh:=(z–1/z)/2 end;
  function tg (z: real): real; begin tg:=sin(z)/cos(z) end;
begin
  read(x); y:=sh(x)*tg(x+1)–sqr(tg(2+sh(x–1))); writeln(y:1:5)
end.
```

11.20. 1+2+3=6

11.23.
```
program многочлены (input, output);
  const n=30;
  type вект = array [0..n] of real;
  var a, b, c: вект; x, y, z, d: real;
  procedure ввод (var v: вект); {ввод вектора}
    var i: integer;
    begin for i:=0 to n do read(v[i]) end;
  function знач (var v: вект; t: real): real;
    {вычисление значения многочлена v в точке t}
    var s: real; i: integer;
    begin s:=v[0]; for i:=1 to n do s:=s*t+v[i]; знач:=s end;
begin
  ввод(a); ввод(b); ввод(c); read(x,y,z);
  d:=(sqr(знач(a,x))– знач (b,y)) / знач (c,x+z);
  writeln(d:1:5)
end.
```

11.29. а)
```
function разл (var X: вектор): integer;
  var y: array[0..99] of 0..1;   {y[j]=1, если j входит в X}
      k, i: integer;
  begin
    for i:=0 to 99 do y[i]:=0;
    k:=0;
    for i:=1 to n do
      begin if y[X[i]]=0 then begin k:=k+1; y[X[i]]:=1 end;
    разл:=k
  end;
```

11.30. а)
```
function F (m, n: неотриц): real;
  function fact (k: неотриц): неотриц;
    var i, p: integer;
    begin p:=1; for i:=2 to k do p:=p*i; fact:=p end;
  begin F:=fact(n)*fact(m)/fact(n+m) end;
```

11.32. Локальные: x, y, z, c; глобальные: vect, integer, index, a, b

11.33. a) 8 true
a 5
6 *
a *

11.40. 2 1 false

11.42.
```
function next: char;
 var c: char;
begin repeat read(c) until c<>' '; next:=c end;
 . . .
k:=0; while next<>'.' do k:=k+1
```

11.46.
```
program integrals (input, output);
  var c, d: real;
  function f1 (x: real): real; begin f1:=arctan(x) end;
  function f2 (x: real): real; begin f2:=sin(exp(10*x)) end;
  function int (function f(x: real): real; a, b: real; n: integer): real;
    var h, s: real; i: integer;
    begin
      h:=(b-a)/n; s:=(f(a)+f(b))/2;
      for i:=1 to n-1 do s:=s+f(a+i*h);
      int:=h*s
    end; {of int}
begin
  read(c,d);
  writeln(int(f1,c,d,20)+int(f2,0,3.1415927,100):1:5)
end.
```

12. РЕКУРСИЯ

12.1. g=1; h=2+h=2+1+h=2+1+0=3

12.2. f(3)=3*f(2)=3*2*f(1)=3*2*1*f(0)= 3*2*1*1=6

12.4.
```
function f2 (n: integer): integer;
begin if n<=1 then f2:=1 else f2:=n*f2(n-2) end;
```

12.6.
```
function pow (x: real; n: integer): real;
begin
  if n=0 then pow:=1 else
    if n<0 then pow:=1/pow(x,abs(n))
           else pow:=x*pow(x,n-1)
end;
```

12.8.
```
function C (m, n: integer): integer;
begin
  if (m=0) or (m=n) then C:=1 else C:=C(m,n-1)+C(m-1,n-1)
end;
```


12.10.

$$f(n)=\begin{cases} n-10 & \text{при } n>100 \\ 91 & \text{иначе} \end{cases}$$

12.11. г)
```
function sum (N: integer): integer;
begin
  if N<10 then sum:=N else sum:=N mod 10 +sum(N div 10)
end;
```

12.12. е)
```
procedure revprint (N: integer);
begin
  if N<10 then write(N) else
    begin
      write(N mod 10)  {сначала вывод последней цифры}
      revprint(N div 10); {затем вывод в обратном порядке
                           всех остальных цифр}
    end
end;
```

12.13.
```
program S (input, output);
 function sum: real;
  var x: real;
 begin
   read(x); {1-е число}
   if x<0 then sum:=0  {конец ввода}
   else sum:=x+sum {сумма всех чисел = 1-е число + сумма остальных}

 end;
begin writeln(sum:1:5) end.
```

12.20.
```
function min (var x: вектор): real;
  function min1 (k: integer): real; {min x[k..n]}
    var m: real;
  begin
    if k=n then min1:=x[n] {min x[n..n]=x[n]}
    else {min x[k..n] = min(x[k], min x[k+1..n])}
      begin  m:=min1(k+1); if x[k]<m then min1:=x[k] else min1:=m end
  end; {of min1}
begin  min:=min1(1)  end;
```

12.23.
```
function Потомок (a, b: имя): boolean;
begin
  if (a=нет) or (b=нет) then Потомок:=false else
    if a=b then Потомок:=true else
      {если отец или мать – потомок, то и ребенок – потомок:}
      if Потомок(a,Отец(b)) then Потомок:=true
                            else Потомок:=Потомок(a,Мать(b))
end;
```

12.25.
```
program formula (input, output);
  function F: integer;
{F считывает из входного файла все символы, образующие
законченную формулу, и вычисляет ее значение как целое}
    var c, op: char;  x, y: integer;
  begin
    read(c);
    if (c>='0') and (c<='9')
      then {цифра есть формула} F:=ord(c)–ord('0')
      else {началась формула вида (x op y),
              где x и y – более простые формулы}
        begin
      {ввести и вычислить x, ввести op, ввести и вычислить y}
          x:=F; read(op); y:=F;
          case op of
            '+': F:=x+y;  '–': F:=x–y;   '*': F:=x*y
          end;
          read(c)  {пропуск ')' в конце формулы}
        end
  end; {of F}
begin  writeln(F)  end.
```

12.30.
```
  {опережающее описание f:}
function f (n: integer): integer;  forward;
function g (m: integer): integer;   {полное описание g}
  begin  if m=0 then g:=1 else g:=f(m–1)+g(m–1)  end;
function f;   {продолжение описания f}
  begin  if n=0 then f:=1 else f:=n+g(n–1)  end;
```

12.31.
```
program formula (input, output);
  function слаг (var c: char): integer;  forward;
  function сумма: integer;
    var c: char;  k: integer;
  begin
    k:=слаг(c);   {1-е слагаемое; в c попадет знак, ')' или точка}
{пока c – знак, продолжать вводить слагаемые}
    while (c='+') or (c='–') do  if c='+' then k:=k+слаг(c) else k:=k–слаг(c);
    сумма:=k
  end;  {of сумма}
  function слаг;
    var k: integer;
  begin
   read(c);
   if c='(' then {началась сумма в скобках → ввести ее и считать символ за ')'}
     begin k:=сумма; read(c) end
   else  {началось число → ввести его и символ за ним}
     begin
       k:=0;  repeat k:=10*k+ord(c)–ord('0'); read(c) until (c<'0') or (c>'9')
     end;
    слаг:=k
  end;   {of слаг}
begin  writeln(сумма)  end.
```

12.34.
```
function path (var L: лабиринт;  н, к: integer): boolean;
 var visit: array[1..n] of 0..1;  {visit[r]=1, если уже были в r}
     j: integer;
 function search (r1, r2: integer): boolean; {есть ли путь из r1 в r2?}
  label 1, 2;
  var r: integer;
 begin
  if r1=r2 then search:=true else
   begin
    search:=false;
  {найти такую комнату r, что: есть коридор из r1 в r;
   в r еще не были; есть путь из r в r2}
    for r:=1 to n do
     begin
      if L[r1,r]=0 then goto 1;  {нет коридора из r1 в r}
      if visit[r]=1 then goto 1;  {в r уже были}
      visit[r]:=1;  {снова r не посещать}
     if search(r,r2) then  {есть путь из r в r2 → есть и путь из r1 в r2}
       begin search:=true; goto 2 end;
   1: {комната r не подошла → проверить следующую}
     end; {for}
   end;  {if}
 2:
 end; {of search}
begin  {тело функции path}
 for j:=1 to n do visit[j]:=0;  visit[н]:=1;
 writeln(search(н,к))
end.
```

13. КОМБИНИРОВАННЫЕ ТИПЫ. ОПЕРАТОР ПРИСОЕДИНЕНИЯ

13.2.
```
б) type время = record час: 0..23;  мин, сек: 0..59 end;
ж) type строка = packed array [1..12] of char;
     ведомость =
      record
        предмет: строка;
        номергруппы: integer;
        дата:  record ч: 1..31;  м: 1..12;  г: integer end;
        студенты: array [1..25] of
                record фам: строка;  зачетка: integer;  оценка: 2..5 end
      end;
```

13.4. в) Правильно: 1) и 2).

13.5.
```
а) a:=b;
б) a.руб:=c.руб; a.коп:=c.коп;
в) less:=(c.руб<b.руб) or (c.руб=b.руб) and (c.коп<b.коп);
г) a:=b;
   if a.коп<99 then a.коп:= a.коп+1
               else begin a.коп:=0; a.руб:= a.руб+1 end;
д) read(b.руб, b.коп)
```

13.7.
```
procedure СамаяВысокая (var C: список);
 var m, i: integer;
begin
  m:=1; for i:=2 to 30 do if C[i].высота>C[m].высота then m:=i;
  writeln(C[m].название)
end;
```

13.8. d:=sqrt(sqr(p1[x]–p2.x)+sqr(p1[y]–p2.y))

13.13. д) begin x[i].a:=1.2; x[i].b[true]:='*' end

13.14.
```
а) time.час:=18; time.мин:=45; time.сек:=10;
б) with time do begin час:=18; мин:=45; сек:=10 end
```

13.16.
```
procedure следсек (var t: время);
begin
  with t do
    if сек<59 then сек:=сек+1 else
      begin
        сек:=0;
        if мин<59 then мин:=мин+1
                  else begin мин:=0; час:=(час+1) mod 24 end
      end
end;
```

13.20. в) Нельзя: 1), 3).

13.21.
```
а) begin p.a:=p.b; d:=2 end;
б) begin q.a:=b; d:=2 end;
в) begin p.a:=p.b; p.c.d:=2 end;
```

13.22. Правильно: б), в), г).

13.23.
```
а) procedure ПД (var p: поляр; var d: декар);
   begin with p, d do begin x:=r*cos(fi); y:=r*sin(fi) end end;
```

13.24.
```
а) procedure печ (var Гр: группа; Бук: char);
    var k: integer;
   begin
    for k:=1 to 25 do
      with Гр[k], деньрожд do
        if фамилия[1]=Бук then
          writeln(фамилия, ' ', число, '.', месяц, '.', год)
   end;
```


14. МНОЖЕСТВЕННЫЕ ТИПЫ

14.1. ж) 2^n различных множеств.

14.2.
```
а) перечислимый тип из имен a, b и c;
б) все входят;
в) [ ], [a], [b], [c], [a, b], [a, c], [b, c], [a, b, c]
```

14.3. type A = set of ДеньНедели; B = set of пн..пт;

14.4. Неправильно описаны типы *точки, данные, М3, М4*.

14.5. Неправильные: г), к), л), м).

14.6. а) [2, 5, 6]; б) [ ]; в) ошибка.

14.7.
```
а) s:=[ ]; б) s:=['a','e','i','o','u'];
в) s:=['0'..'9'];      г) s:=[succ(c)..pred(d)]
```

14.8. Правильно: б).

14.9.
```
г) Допустимы: 2), 3), 4).
ж) Правильно: 1), 2), 4).
```

14.10. а), г), е), к) – false; б), в), д), ж), з), и) – true; л), м) – ошибка.

14.12.
```
function счет (s: строка): integer;
 var i, k: integer;
begin
  k:=0; for i:=1 to 100 do if s[i] in ['0'..'9', '+', '–', '*'] then k:=k+1;
  счет:=k
end;
```

14.13.
```
function ЧислоДней (m: месяц): integer;
begin
  if m=2 then ЧислоДней:=28 else
   if m in [4, 6, 9, 11] then ЧислоДней:=30 else ЧислоДней:=31
end;
```

14.15.
```
а)  function card (A: Lat): integer;
        var k: integer;  c: char;
    begin k:=0; for c:= 'a' to 'z' do if c in A then k:=k+1; card:=k end;
```

14.17. Правильно: г), д).

14.18. г) Недопустимо: 1).

14.19. а) [1..5]; б) [ ]; в) [1,3,5]; г) [1..8]; д) [3..6]; е) [1,2]; ж) [1..5]; з) [2,4]; и) [ ]; к) [4]; л) [ ]; м) [ ]

14.20. а) [3..5, 7..10, 13]

14.21. а) [ ]; б) A*B

14.23. x:=[8..22]; y:=[11,13,17,19]; z:=x–y

14.25. а) B:=A+[x]; б) B:=A–[x]

14.26. а)
```
function digits (n: натур): натур;
  var sd: set of 0..9; d: 0..9; k: integer;
 begin
{выделение цифр из n и запись их в множество sd:}
  sd:=[ ]; repeat d:=n mod 10; sd:=sd+[d]; n:=n div 10 until n=0;
{подсчет числа элементов в sd:}
  k:=0; for d:=0 to 9 do if d in sd then k:=k+1;
  digits:=k
 end;
```

14.27. а)
```
program ПервыеВхождения (input, output);
  var let: set of 'a'..'z'; c: char;
 begin
  let:=[ ]; {множество букв в рассмотренной части текста}
  read(c);
  while c<>'.' do
   begin
    if not(c in let) then {1-е вхождение}begin write(c); let:=let+[c] end;
    read(c)
   end;
  writeln
 end.
```

15. ФАЙЛОВЫЕ ТИПЫ

15.2. Недопустим тип С.

15.3. Недопустимы: а), в), г).

15.5. а) 2 3; б) 1 2; в) 1 2; г) 1 2

15.6. Ошибки: 1) файловый параметр *w* описан не как параметр-переменная; 2) неправильно обрабатывается пустой файл *w* (чтение из пустого файла).

15.7.
```
function отриц (var s: серия): real;
  var sum, x: real;
 begin
  reset(s); sum:=0;
  while not eof(s) do begin read(s,x); if x<0 then sum:=sum+x end;
  отриц:=sum
 end;
```

15.9. а)
```
function eq (var w1, w2: слово): boolean;
  var c1, c2: char; ok: boolean;
 begin
  reset(w1); reset(w2); ok:=true;
  while not eof(w1) and not eof(w2) and ok do
   begin read(w1,c1); read(w2,c2); ok:=c1=c2 end;
  eq:=ok and eof(w1) and eof(w2)
 end;
```


15.10. а)
```
function CA (var Б: библиотека): integer;
  var k, j: integer;
 begin
  k:=0;
  for j:=1 to 1000 do
   begin
    reset(Б[j]); if not eof(Б[j]) then if Б[j]↑='A' then k:=k+1
   end;
  CA:=k
 end;
```

15.13. а)
```
function упор (var r: ряд): boolean;
  var x, y: integer; {x и y –очередные элементы файла}
      ok: boolean;   {=true, пока x<y}
 begin
  reset(r); read(r,y); ok:=true;
  while not eof(r) and ok do begin x:=y; read(r,y); ok:=x<y end;
  упор:=ok
 end;
```

15.18. а) ошибка; б) 1 2; в) 1 2; г) ошибка; д) 2; е) 1 3

15.19.
```
procedure цифры (var s: строка; var t: текст);
  var i: integer;
 begin
  rewrite(t); for i:=1 to 100 do if s[i] in ['0'..'9'] then write(t,s[i])
 end;
```

15.21.
```
procedure присв (var f, g: FB);
  var b: boolean;
 begin
  reset(g); rewrite(f); while not eof(g) do begin read(g,b); write(f,b) end
 end;
```

15.26.
```
procedure присв (var f, g: Ф);
 begin
  reset(g); rewrite(f);
  while not eof(g) do begin write(f,g↑); get(g) end
 end;
```

15.28. а)
```
function less (var f: FR): integer;
  var k: integer; x, s: real;
 begin
 {подсчет среднего арифметического:}
  reset(f); k:=0; s:=0; repeat read(f,x); k:=k+1; s:=s+x until eof(f);
  s:=s/k;
 {новый просмотр f и подсчет элементов < s :}
  reset(f); k:=0; repeat read(f,x); if x<s then k:=k+1 until eof(f);
  less:=k
 end;
```

15.32. a)
```
procedure add1 (var t: текст; c: char);
  var d: char;  s: текст; {вспомогательный файл}
begin
  {копирование t в s:}
 reset(t); rewrite(s); while not eof(t) do begin read(t,d); write(s,d) end;
  {запись с и s в t:}
 reset(s); rewrite(t);
 write(t,c); while not eof(s) do begin read(s,d); write(t,d) end
end;
```

15.37.
```
procedure triangle (var t: text);
  var c, d: char;
begin
 rewrite(t);
 for d:= '1' to '9' do begin for c:= '1' to d do write(t,d); writeln(t) end
end;
```

15.39. Правильно: а).

15.40. k=2

15.41. Все правильные.

15.42. a)
```
function empty (var t: text): integer;
  var k, d: integer;
begin
 reset(t); k:=0;
 while not eof(t) do begin if eoln(t) then k:=k+1; readln(t) end;
 empty:=k
end;
```

15.44. a)
```
function count (var t: text): integer;
  var k: integer;  c: char;
begin
 reset(t); k:=0;
 while not eof(t) do {цикл по строкам}
  begin
   read(t,c); if c='d' then k:=k+1; {чтение и анализ 1-го символа}
   readln(t)  {пропуск остатка строки}
  end;
 count:=k
end;
```

15.46.
```
procedure присв (var t1, t2: text);
  var c: char;
begin
  reset(t2); rewrite(t1);
  while not eof(t2) do
    if eoln(t2) then begin readln(t2); writeln(t1) end
                else begin read(t2,c); write(t1,c) end
end;
```


15.52.
```
procedure зап (var l: список; var t: text);
  var i: integer;
begin rewrite(t); for i:=1 to 100 do writeln(t, l[i]) end;
```

15.53.
```
function max (var t: text): real;
  var m, x: real;
begin
  reset(t); readln(t,m);
  while not eof(t) do begin readln(t,x);  if x>m then m:=x end;
  max:=m
end;
```

15.57.
```
program formatting (BOOK);
  const d=60; {длина «новых» строк}
  var BOOK, COPY: text;  c: char;  k: integer;
begin
{перепись BOOK во внутр. файл COPY с удалением «концов строк»:}
  reset(BOOK);  rewrite(COPY);
  while not eof(BOOK) do
    if eoln(BOOK) then readln(BOOK)
                  else begin read(BOOK,c);  write(COPY,c) end;
{перенос текста из COPY в BOOK с форматированием:}
  reset(COPY);  rewrite(BOOK);
  k:=0; {k – порядковый номер символа в текущей строке}
  while not eof(COPY) do
   begin
     read(COPY,c); write(BOOK,c);  k:=k+1;
     if (c='.') or (k=d) then begin  writeln(BOOK);  k:=0 end
   end
end.
```

16. ССЫЛОЧНЫЕ ТИПЫ. СПИСКИ

16.2. Неправильно описаны типы *D* и *F*.

16.4. 4 3

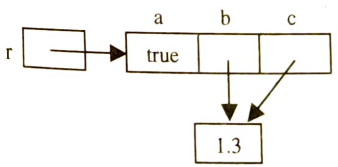


Рис. 26

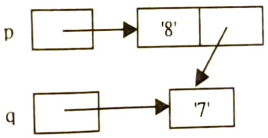


Рис. 27

16.5. а) *r* – ссылка на запись типа *D*; *r*↑ – сама запись; *r*↑.*b* – поле *b* этой записи, значением которого является ссылка на вещественную переменную со

значением 2.7; $r\uparrow.b\uparrow$ – сама вещественная переменная.

б) См. рис. 26

16.6. Неправильные: б), д), е), ж), и), к), л), м).

16.8. См. рис. 27

16.9. Неправильно описаны типы *A1* и *D1*.

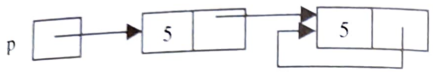


Рис. 28

16.10. г) См. рис. 28.

16.11. б)
```
type C = ↑char;  A = ↑R;  R = record f1: C; f2: A end;
var p, q: A;  x: C;
...
new(p);  new(q);
new(x);  x↑:='a';  p↑.f1:=x;  p↑.f2:=q;
new(x);  x↑:='b';  q↑.f1:=x;  q↑.f2:=nil
```

16.12. 1) dispose(р.кон); р.кон:= р.нач

16.15. б)
```
function neg1 (var x: вектор): ссылка;
  var i: integer;  t: boolean;
begin
  i:=0; repeat i:=i+1; t:=x[i] ↑<0 until t or (i=100);
  if t then neg1:=x[i] else neg1:=nil
end;
```

16.16. б)
```
function элем (var T: текст; i, j: integer; var c: char): boolean;
  begin
   if T[i]=nil then элем:=false else begin элем:=true; c:=T[i]↑[j] end
  end;
```

16.17. а) E:=p↑.элем;
б) t:=p↑.след=nil;
в) p:=p↑.след;
г) if L=nil then t:=false else t:=L↑.след<>nil

(*Замечание*: решение t:=(L<>nil) and (L↑.след<>nil) недопустимо из-за ошибки в L↑.след при L=nil.)

16.18. {списки без заглавного звена}

а)
```
function послед (L: список): ТЭ;
  var p: список; {ссылка на очередное звено}
begin
  p:=L; while p↑.след<>nil do p:=p↑.след;   {поиск звена с nil}
  послед:=p↑.элем
end;
```


б)
```
function длина (L: список): integer;
  var p: список;  k: integer;
begin
  k:=0; p:=L;  while p<>nil do begin k:=k+1;  p:=p↑.след end;
  длина:=k
end;
```

д)
```
procedure замена (L: список;  E1, E2: ТЭ);
  var p: список;
begin
  p:=L;
  while p<>nil do
   begin if p↑.элем=E1 then p↑.элем:=E2; p:=p↑.след end;
end;
```

ж)
```
procedure замена (L: список;  E: ТЭ);
  var pmin, p: список;
begin
   {поиск pmin – ссылки на звено с min:}
  pmin:=L;  p:=L↑.след;
  while p<>nil do
   begin if p↑.элем<pmin↑.элем then pmin:=p;  p:=p↑.след end;
  pmin↑.элем:=E  {замена min на E}
end;
```

и)
```
function упор (L: список): boolean;
  var p, q: список; {ссылки на пару соседних звеньев}
  ok: boolean;
begin
  p:=L;  q:=L↑.след;  ok:=true;
  while (q<>nil) and ok do
   begin ok:=p↑.элем <= q↑.элем;  p:=q;  q:=q↑.след end;
  упор:=ok
end;
```

16.20. а)
```
{список с заглавным звеном}
function newlist (var f: файл): список;
  var x: ТЭ;  L, p, q: список; {L – список из уже созданных
                                звеньев, p – последнее из них}
begin
  reset(f);
  new(L); {создать заглавное звено}
  p:=L;
  while not eof(f) do
   begin
     read(f,x);
     new(q); q↑.элем:=x; {создать звено с x}
     p↑.след:=q; {добавить его вслед за p}
     p:=q  {объявить его последним}
```

```
      end;
      p↑.след:=nil; {записать nil в последнее звено}
      newlist:=L
    end;
в) {список без заглавного звена}
    function newlist (var x: массив): список;
      var L, p: список;  k: integer;
    begin
      L:=nil;
      for k:=50 downto 1 do
       begin
        new(p); p↑.элем:=x[k]; {создать звено с x[k]}
        p↑.след:=L;  L:=p  {добавить его в начало L}
       end;
      newlist:=L
    end;
```

16.22.

```
{списки без заглавного звена}
б) procedure вставка (L: список;  E: ТЭ);
     var p: список;
   begin new(p); p↑.элем:=E; p↑.след:=L↑.след;  L↑.след:=p end;

в) procedure вставка (var L: список;  E: ТЭ);
     var p, q: список;
   begin
     new(p);  p↑.элем:=E;  p↑.след:=nil;  {новое звено с E}
     if L=nil then L:=p  {был пустой список}
     else
      begin {поиск q – последнего звена}
        q:=L; while q↑.след<>nil do q:=q↑.след;
        q↑.след:=p  {вставка p за q}
      end
   end;

е) procedure вставка (L: список;  E, E1: ТЭ);
     var p, q: список;  eq: boolean;
   begin
   {поиск p – звена с E:}
     p:=L; eq:=false;
     while (p<>nil) and not eq do
       if p↑.элем=E then eq:=true else p:=p↑.след;
     if eq then {нашли --> вставка E1 перед E}
      begin
   {внимание – трюк: запись E1 в звено p вместо E
    и вставка E за звеном p:}
      p↑.элем:=E1;  new(q);  q↑.элем:=E; q↑.след:=p↑.след;  p↑.след:=q
      end
   end;
```


16.23.

```
{списки без заглавного звена}
б) procedure удалить (L: список);
     var p, q: список;
   begin
    if L<>nil then if L↑.след<>nil then
     begin
       p:= L↑.след;  {ссылка на 2-е звено}
       L↑.след:=p↑.след;  {удаление 2-го звена из цепочки}
       dispose(p)  {уничтожение 2-го звена}
     end
   end;

д) procedure удалить (var L: список);
     var p, q: список;
   begin
    if L↑.след=nil then  {всего одно звено}
     begin dispose(L); L:=nil end
    else
     begin
        { найти предпоследнее (p) и последнее (q) звенья: }
       p:=L;  q:=p↑.след;
       while q↑.след<>nil do begin p:=q;  q:=q↑.след end;
       dispose(q);  {уничтожить последнее звено}
       p↑.след:=nil {предпоследнее звено стало последним}
     end
   end;
```

16.25.

```
program reverse (input, output);
  type список = ↑звено;
       звено = record элем: char;  след: список end;
  var L, p: список;  c: char;
begin
{ввод символов и запись их в обратном порядке в список L}
  L:=nil;  {ссылка на построенную часть списка}
  read(c);
  while c<>'.' do begin new(p); p↑.элем:=c; p↑.след:=L; L:=p; read(c) end;
{вывод симолов из L:}
  while L<>nil do begin write(L↑.элем);  L:=L↑.след  end;
  writeln
end.
```

16.30.

```
{списки без заглавных звеньев}
a) function memb (L: список;  E: ТЭ): boolean;
     begin
       if L=nil then memb:=false else
        if L↑.элем=E then memb:=true
                     else memb:=memb(L↑.след, E)
     end;
```

з)
```
procedure delete (var L: список;  E: ТЭ);
  var p: список;
begin
  if L<>nil then
   if L↑.элем=E then  {удалить 1-е звено}
    begin p:=L; L:=L↑.след; dispose(p) end
   else delete(L↑.след,E)  {удалить E из «хвоста» списка и
     записать в 1-е звено ссылку на на измененный «хвост»}
end;
```

д)
```
procedure insert (L: список;  E, E1: ТЭ);
  var p: список;
begin
  if L<>nil then
   if  L↑.элем<>E then insert(L↑.след,E,E1) else
    begin new(p);  p↑.элем:=E1;  p↑.след:= L↑.след;  L↑.след:=p  end
end;
```

16.33. б)
```
{список без заглавного звена}
procedure firstletters (L: список);
begin
  while L<>nil do
   begin if L↑.элем<>nil then write(L↑.элем↑.буква); L:=L↑.след end;
  writeln
end;
```

16.41. а)
```
function single (L: список): boolean;
begin if L=nil then single:=false else single:=L↑.след=L end;
```

е)
```
procedure delete (var L: список);
  var  q, p: список;
begin
  if L↑.след=L then  {список из одного звена}
   begin dispose(L); L:=nil end
  else
    begin
     q:=L; p:=L↑.след; {поиск предпослед. (q) и послед.(p) звеньев}
     while p↑.след<>L do begin q:=p; p:=p↑.след end;
     q↑.след:=L; dispose(p)   {удаление последнего звена}
    end
end;
```

16.43. а)
```
procedure build1 (var L: список2;  E: ТЭ2);
  var p: список2;
begin
  new(p); p↑.элем:=E; p↑.пред:=nil; p↑.след:=nil;
  new(L);  L↑.пред:=p;  L↑.след:=p  {заглавное звено}
end;
```


б)
```
procedure revprint (L: список2);
  var p: список2;
begin
   p:=L↑.пред; {последнее звено}
   while p<>nil do begin write(p↑.элем);  p:=p↑.пред end;
   writeln
end;
```

17. ОЧЕРЕДИ, СТЕКИ, ДВОИЧНЫЕ ДЕРЕВЬЯ

17.1. а)
```
type очередь =
    record  нач, кон: 0..n;  мас: array [1..n] of ТЭО  end;
procedure ОЧИСТОЧ (var Q: очередь);
 begin Q.нач:=1; Q.кон:=0 end;
function ПУСТОЧ (var Q: очередь): boolean;
 begin ПУСТОЧ:=Q.кон=0 end;
procedure ВОЧЕРЕДЬ (var Q: очередь;  x: ТЭО);
   var i: integer;
  begin
   with Q do
    begin
     if кон=n then {очередь у правого края}
      if нач=1 then {занят весь мас} ОШИБКА(1)
      else {сдвиг к левому краю}
        for i:=нач to кон do мас[i–нач+1]:=мас[i];
      кон:=кон+1;  мас[кон]:=x
    end
  end;
 procedure ИЗОЧЕРЕДИ (var Q: очередь;  var x: ТЭО);
  begin
   if ПУСТОЧ(Q) then ОШИБКА(2);
   with Q do
     begin
      x:=мас[нач];
      if нач=кон then {был один элемент} ОЧИСТОЧ(Q)
                 else нач:=нач+1
     end
  end;
```

17.2. а)
```
procedure вывод (var f: FR;  a, b: real);
   var Q1, Q2: очередь; {ТЭО=real}  x: real;
begin
  reset(f);  ОЧИСТОЧ(Q1);  ОЧИСТОЧ(Q2);
  {вывод чисел < a, запись в Q1 чисел из [a,b] и
   запись в Q2 чисел > b:}
  while not eof(f) do
```

```
    begin
      read(f,x);
      if x<a then write(x:1:5,' ') else
        if x<=b then ВОЧЕРЕДЬ(Q1,x) else ВОЧЕРЕДЬ(Q2,x)
    end;
    {вывод чисел из Q1 и Q2:}
    while not ПУСТОЧ(Q1) do
      begin ИЗОЧЕРЕДИ(Q1,x); write(x:1:5,' ') end;
    while not ПУСТОЧ(Q2) do
      begin ИЗОЧЕРЕДИ(Q2,x); write(x:1:5,' ') end;
    writeln
  end;
```

17.4. в)

```
function formula: integer;
  var S: стек;  c, op, x, y: char;
begin
  ОЧИСТЕК(S);
  read(c);
  while c<>'.' do
    begin
      {обработка очередного символа текста:}
    if c in ['0'..'9', 'M', 'm'] then ВСТЕК(S,c) {цифры, M и m}
    else
      if c=')' then   {конец формулы вида op(x,y)}
        begin
    {в стеке находится тройка op x y, она удаляется из стека;
    выполнение операции op и запись результата в стек:}
          ИЗСТЕКА(S,y);  ИЗСТЕКА(S,x);  ИЗСТЕКА(S,op);
          case op of
            'M' {max}: if x>y then c:=x else c:=y;
            'm' {min}:  if x<y then c:=x else c:=y
          end;
          ВСТЕК(S,c)
        end;
      {символы '(' и ',' игнорируются}
    read(c)
  end; {of while}
    {в стеке осталась цифра – значение всей формулы}
  ИЗСТЕКА(S,c);  formula:=ord(c)–ord('0')
end;
```

17.7. б)

```
function count (T: дерево;  E: ТЭД): integer;
  var S: стек; {ТЭС=дерево}  k: integer;
begin
  ОЧИСТЕК(S);
  k:=0; {число вершин с E}
  while T<> nil do
```



```
    begin {T – ссылка на очередную вершину}
      if T↑.элем=E then k:=k+1;
        {переход к следующей вершине:}
      if T↑.лев <> nil then {есть ветвь влево}
        begin
            {правая ветвь, если есть, → стек}
          if T↑.прав <> nil then ВСТЕК(S,T↑.прав);
          T:=T↑.лев   {идти  влево}
        end
      else
      if T↑.прав <> nil then {идти вправо:} T:=T↑.прав
      else
    {нет обеих ветвей —>  взять ветвь из стека и идти по ней}
        if not ПУСТЕК(S) then ИЗСТЕКА(S,T)
                         else T:=nil {конец просмотра}
    end; {of while}
    count:=k
  end;
```

ж)

```
procedure levels (T: дерево);
  var Q: очередь;  {ТЭО=дерево}  n, k, k1: integer;
begin
  if T <> nil then
    begin
      ОЧИСТОЧ(Q); ВОЧЕРЕДЬ (Q,T);  {корень → стек}
      n:=1;  {номер уровня}
      k:=1; {число вершин на этом уровне}
      repeat {цикл по уровням дерева}
    {в очереди Q находятся ссылки на все (k) вершины
     n-го уровня дерева; они считываются и выводятся;
     их дочерние вершины, т.е. вершины (n+1)-го уровня.}
        write (n, '-й уровень: ');
        k1:=0;
        for k:=1 to k do
          begin
            ИЗОЧЕРЕДИ(Q,T); write (T↑.элем, ' ');
            if T↑.лев <> nil then
              begin ВОЧЕРЕДЬ(Q,T↑.лев); k1:=k1+1 end;
            if T↑.прав <> nil then
              begin ВОЧЕРЕДЬ(Q,T↑.прав); k1:=k1+1 end
          end;
        writeln; {конец вывода n-го уровня}
        n:=n+1; k:=k1 {на следующий уровень}
      until k=0
    end { T<> nil }
  end;
```

17.8. б)
```
function count (T: дерево;  E: ТЭД): integer;
  var k: integer;
 begin
  if T=nil then count:=0 else
   begin
    if T↑.элем=E then k:=1 else k:=0;
    count:=k+count(T↑.лев,E)+count(T↑.прав,E)
   end
 end;
```
и)
```
function heigth (T: дерево): integer;
  var h1, h2: integer;
 begin
  if T=nil then heigth:=0 else
   begin
      {высоты левого и правого поддеревьев:}
    h1:=heigth(T↑.лев);  h2:=heigth (T↑.прав);
      {взять максимальную + учесть корень:}
    if h1>h2 then heigth:=h1+1 else heigth:=h2+1
   end
 end;
```

17.11.
```
procedure copy (T: дерево; var T1: дерево);
begin
 if T=nil then T1:=nil else
  begin
   new(T1); T1↑.элем:=T↑.элем;  {копия корня}
    {копии левого и правого поддеревьев:}
   copy(T↑.лев,T1↑.лев); copy(T↑.прав,T1↑.прав)
  end
end;
```

17.14. а)
```
function path (T: дерево; S: integer): boolean;
  {используется обобщение функции: path(nil,S)≡(S=0)}
 begin
  if T=nil then path:=S=0 else
   if T↑.элем>S then path:=false else
    if path(T↑.лев,S–T↑.элем) then path:=true
     else path:= path(T↑.прав,S–T↑.элем)
 end;
```

17.15. а)
```
function value (T: дерево): integer;
  var x, y: integer;
 begin
  if T↑. элем in ['0'..'9'] then value:=ord(T↑. элем)–ord('0')
  else
   begin
    x:=value(T↑.лев); y:=value(T↑.прав);
    case T↑.элем of
```



```
     '+': value:=x+y;  '–': value:=x–y;  '*': value:=x*y
    end
   end
 end;
```

17.16. а)
```
procedure осв (var T: дерево);
  {вспомогательная процедура: освободить память,
   занимаемую деревом T, и T:=nil}
 begin
  if T <> nil then
  begin осв(T↑.лев); осв(T↑.прав); dispose(T); T:=nil end
 end;
 procedure упростить (var T: дерево);
  var T1, T2: дерево;
 begin
  if T↑.элем in ['+', '–', '*'] then
   begin
     {сначала упростить операнды формулы:}
    упростить(T↑.лев);  упростить(T↑.прав);
    T1:=T↑.лев;  T2:==T↑.прав;
     {упростить саму формулу:}
    if (T↑.элем='+') and (T1↑.элем='0') or
       (T↑.элем='*') and (T1↑.элем='1')
     then { (0+φ), (1*φ) → φ }
      begin dispose(T1); dispose(T); T:=T2 end
    else
    if (T↑.элем in ['+', '–']) and (T2↑.элем='0') or
       (T↑.элем='*') and (T2↑.элем='1')
     then { (φ+0), (φ–0), (φ*1) → φ }
      begin dispose(T2); dispose(T); T:=T1 end
    else
    if (T↑.элем='*') and ((T1↑.элем='0') or (T2↑.элем='0'))
     then { (0*φ), (φ*0) → 0 }
      begin осв(T↑.лев); осв(T↑.прав); T↑.элем:='0' end
   end
 end;
```

17.17. а)
```
function memb (T: дерево: E: ТЭД): boolean;
  var eq: boolean;
 begin
  eq:=false;
  while (T<>nil) and not eq do
   if E=T↑.элем then eq:=true else
    if E< T↑.элем then T:= T↑.лев else T:= T↑.прав;
   memb:=eq
 end;
```