**ЗАДАНИЯ**

**3 тура дистанционной олимпиады по программированию**

**для 10-11 классов**

Галимарданов Альберт Асхатович (MrTrollolo1337@gmail.com)

Адрес: Россия, Республика Башкортостан, Агидель, Цветочный бульвар, 4

МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ "СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА №1" ГОРОДСКОГО ОКРУГА ГОРОД АГИДЕЛЬ РЕСПУБЛИКИ БАШКОРТОСТАН

Решение произведено в программе PascalABC.NET версии 3.3.

**Задание 1**

Разработать программу, которая заполняет и выводит на экран таблицу истинности функции **(A \/ B) /\¬C**.

**program** qq;

**var** f:real;

**begin**

write('Значение функции F = ');

readln(f);

**if** f=0 **then begin**//выводим таблицу для всех ложных значений

writeln('|A|B|C|F|');

writeln('|0|0|0|0|');

writeln('|0|0|1|0|');

writeln('|0|1|1|0|');

writeln('|1|0|1|0|');

writeln('|1|1|1|0|');

**end**;

**if** f=1 **then begin**//выводим таблицу для всех истинных значений

writeln('|A|B|C|F|');

writeln('|0|1|0|1|');

writeln('|1|0|0|1|');

writeln('|1|1|0|1|');

**end**;

**if** (f<>1)**and**(f<>0)**then** writeln('Только 0 или 1');

**end**.

Входные данные

Значение функции F = 1

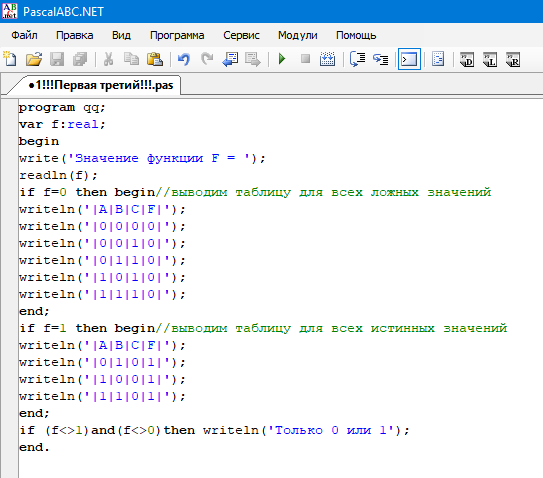
Выходные данные

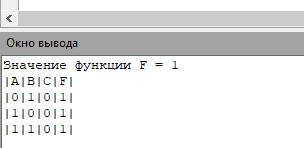
|A|B|C|F|

|0|1|0|1|

|1|0|0|1|

|1|1|0|1|





**Задание 2**

Разработать программу, которая выполняет классификации четырехугольников по величине углов и длине сторон для вводимых с клавиатуры значений.

**program** qq;

**var** ab,bc,cd,ad,abc,bad,bcd,adc,i,k,p,x,pu,ps:integer;

dannoe,s:string;

**begin**

writeln('Вводите данные, пока программа не выведет вам ответ');

writeln('Программа работает для углов BAD, ABC, BCD, ADC и сторон AB, BC, CD, AD(заглавными буквами и на английском языке)');

**for** i:=1 **to** 8 **do begin**

k:=0;

s:='';

readln(dannoe);

k:=pos('торона',dannoe);

**if** k>0 **then begin**

k:=0;

//'работа со сторонами'

k:=pos('AB',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,ab,p);

**end**;

k:=pos('BC',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,bc,p);

**end**;

k:=pos('CD',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,cd,p);

**end**;

k:=pos('AD',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,ad,p);

**end**;

**end else begin**

k:=0;

//'работа с углами'

k:=pos('ABC',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,abc,p);

**end**;

k:=pos('BAD',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,bad,p);

**end**;

k:=pos('BCD',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,bcd,p);

**end**;

k:=pos('ADC',dannoe);

**if** k>0 **then begin** k:=pos('=',dannoe);

**for** x:=k+1 **to** length(dannoe) **do**

s:=s+dannoe[x];

k:=0;

val(s,adc,p);

**end**;

**end**;

**if** ((bad = 90) **and** (abc = 90))**and**((bcd = 90) **or** (adc = 90))**and**((ab = bc)**or**(ab = ad)**or**(cd = ad)**or**(cd = bc))**then begin** writeln('Квадрат');**break**;**end**;

**if** ((bad = 90) **or** (abc = 90))**and**((bcd = 90) **and** (adc = 90))**and**((ab = bc)**or**(ab = ad)**or**(cd = ad)**or**(cd = bc))**then begin** writeln('Квадрат');**break**;**end**;

**if** ((bad = 90) **and** (abc = 90))**and**((bcd = 90) **or** (adc = 90))**and**((ab <> bc)**or**(ab <> ad)**or**(cd <> ad)**or**(cd <> bc))**then begin** writeln('Прямоугольник');**break**;**end**;

**if** ((bad = 90) **or** (abc = 90))**and**((bcd = 90) **and** (adc = 90))**and**((ab <> bc)**or**(ab <> ad)**or**(cd <> ad)**or**(cd <> bc))**then begin** writeln('Прямоугольник');**break**;**end**;

**if** ((bad=bcd) **and** (abc=180-bcd)) **and** ((ab=bc)**and**(bc=cd)) **or** ((ab=bc)**and**(bc=ad)) **or** ((cd=ad)**and**(ad=ab)) **or** ((cd=ad)**and**(ad=bc)) **then begin** writeln('Ромб');**break**;**end**;

**if** ((bad=bcd) **and** (adc=180-bcd)) **and** ((ab=bc)**and**(bc=cd)) **or** ((ab=bc)**and**(bc=ad)) **or** ((cd=ad)**and**(ad=ab)) **or** ((cd=ad)**and**(ad=bc)) **then begin** writeln('Ромб');**break**;**end**;

**if** ((adc=bad) **and** (bcd=180-bad)) **and** ((ab=bc)**and**(bc=cd)) **or** ((ab=bc)**and**(bc=ad)) **or** ((cd=ad)**and**(ad=ab)) **or** ((cd=ad)**and**(ad=bc)) **then begin** writeln('Ромб');**break**;**end**;

**if** ((adc=bad) **and** (abc=180-bad)) **and** ((ab=bc)**and**(bc=cd)) **or** ((ab=bc)**and**(bc=ad)) **or** ((cd=ad)**and**(ad=ab)) **or** ((cd=ad)**and**(ad=bc)) **then begin** writeln('Ромб');**break**;**end**;

**if** ((bad=bcd) **and** (abc=180-bcd)) **and** (((ab<>bc)**and**(ab=cd)) **or** ((bc<>cd)**and**(bc=ad))**or**((cd<>ad)**and**(cd=ab)) **or** ((ad<>ab)**and**(ad=bc))) **then begin** writeln('Параллелограмм');**break**;**end**;

**if** ((bad=bcd) **and** (adc=180-bcd)) **and** (((ab<>bc)**and**(ab=cd)) **or** ((bc<>cd)**and**(bc=ad))**or**((cd<>ad)**and**(cd=ab)) **or** ((ad<>ab)**and**(ad=bc))) **then begin** writeln('Параллелограмм');**break**;**end**;

**if** ((adc=bad) **and** (bcd=180-bad)) **and** (((ab<>bc)**and**(ab=cd)) **or** ((bc<>cd)**and**(bc=ad))**or**((cd<>ad)**and**(cd=ab)) **or** ((ad<>ab)**and**(ad=bc))) **then begin** writeln('Параллелограмм');**break**;**end**;

**if** ((adc=bad) **and** (abc=180-bad)) **and** (((ab<>bc)**and**(ab=cd)) **or** ((bc<>cd)**and**(bc=ad))**or**((cd<>ad)**and**(cd=ab)) **or** ((ad<>ab)**and**(ad=bc))) **then begin** writeln('Параллелограмм');**break**;**end**;

**if** ((abc=bad)**and**((bcd=180-abc)**or**(adc=180-abc)))**or**((abc=bcd)**and**((adc=180-abc)**or**(bad=180-abc)))**or**((bcd=adc)**and**((bad=180-adc)**or**(abc=180-adc)))**or**((adc=bad)**and**((bcd=180-adc)**or**(abc=180-adc)))**then begin** writeln('Равнобедренная трапеция');**break**;**end**;

**if** ((ab=bc)**or**(bc=cd)**or**(cd=ad)**or**(ab=ad))**and**(((bad=90)**and**(bad=abc))**or**((abc=90)**and**(abc=bcd))**or**((bcd=90)**and**(bcd=adc))**or**((adc=90)**and**(adc=bad))**or**((bad=90)**and**(bad=bcd))**or**((adc=90)**and**(adc=abc)))**then begin** writeln('Квадрат');**break**;**end**;

**if** ((ab=bc)**and**(bc=cd)**and**(cd=ad))**and**((bad=90)**or**(abc=90)**or**(bcd=90)**or**(adc=90))**then begin** writeln('Квадрат');**break**;**end**;

**if** ((ab=bc)**and**(bc=cd)**and**(cd=ad))**and**((bad<>90)**or**(abc<>90)**or**(bcd<>90)**or**(adc<>90))**then begin** writeln('Ромб');**break**;**end**;

**if** (bad=90)**and**(abc=90)**and**(bcd=180-adc)**then begin** writeln('Прямоугольная трапеция');**break**;**end**;

**if** (abc=90)**and**(bcd=90)**and**(adc=180-bad)**then begin** writeln('Прямоугольная трапеция');**break**;**end**;

**if** (bcd=90)**and**(adc=90)**and**(bad=180-abc)**then begin** writeln('Прямоугольная трапеция');**break**;**end**;

**if** (adc=90)**and**(bad=90)**and**(bcd=180-abc)**then begin** writeln('Прямоугольная трапеция');**break**;**end**;

**if** (bad=180-abc)**and**(bcd=180-adc)**and**(bad<>bcd)**then begin** writeln('Трапеция');**break**;**end**;

**if** (abc=180-bcd)**and**(adc=180-bad)**and**(bad<>bcd)**then begin** writeln('Трапеция');**break**;**end**;

**if** (bcd=180-adc)**and**(bad=180-abc)**and**(bad<>bcd)**then begin** writeln('Трапеция');**break**;**end**;

**if** (adc=180-bad)**and**(abc=180-bcd)**and**(bad<>bcd)**then begin** writeln('Трапеция');**break**;**end**;

**if** (bad>0)**and**(abc>0)**and**(bcd>0)**and**(adc>0)**and**(ab>0)**and**(bc>0)**and**(cd>0)**and**(ad>0)**then** writeln('Обычный четырехугольник');

**end**;

**end**.

Входные данные

Вводите данные, пока программа не выведет вам ответ

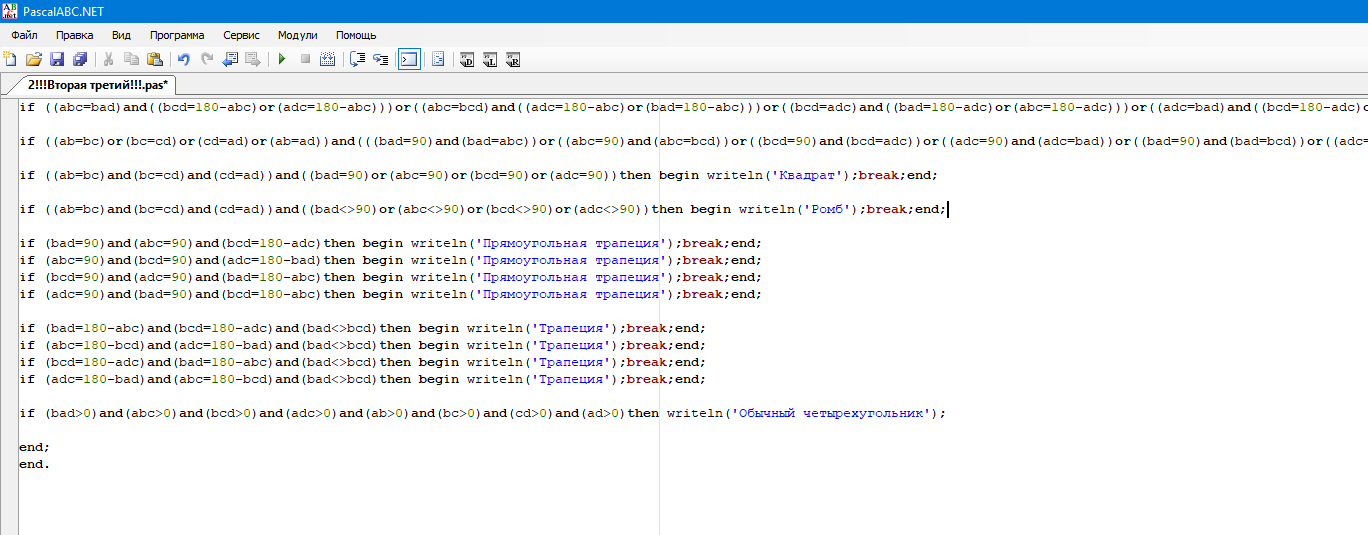
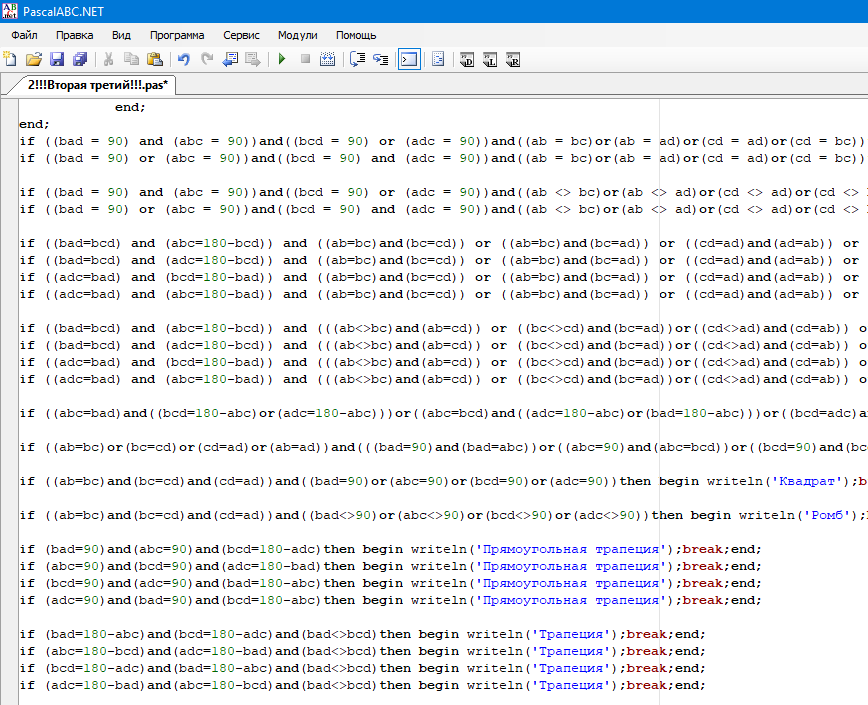
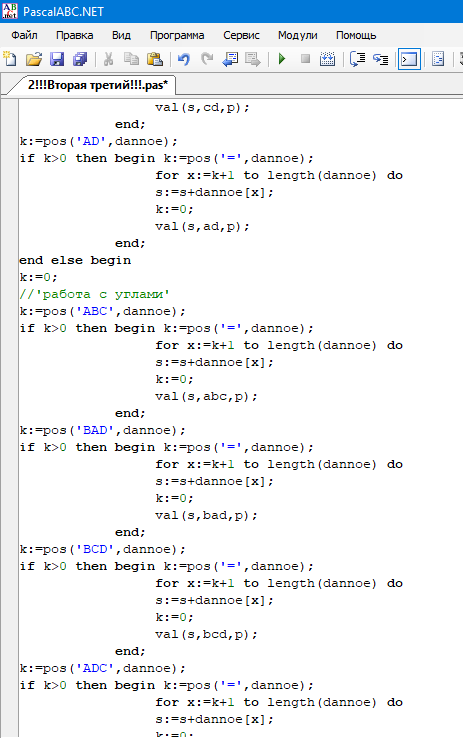
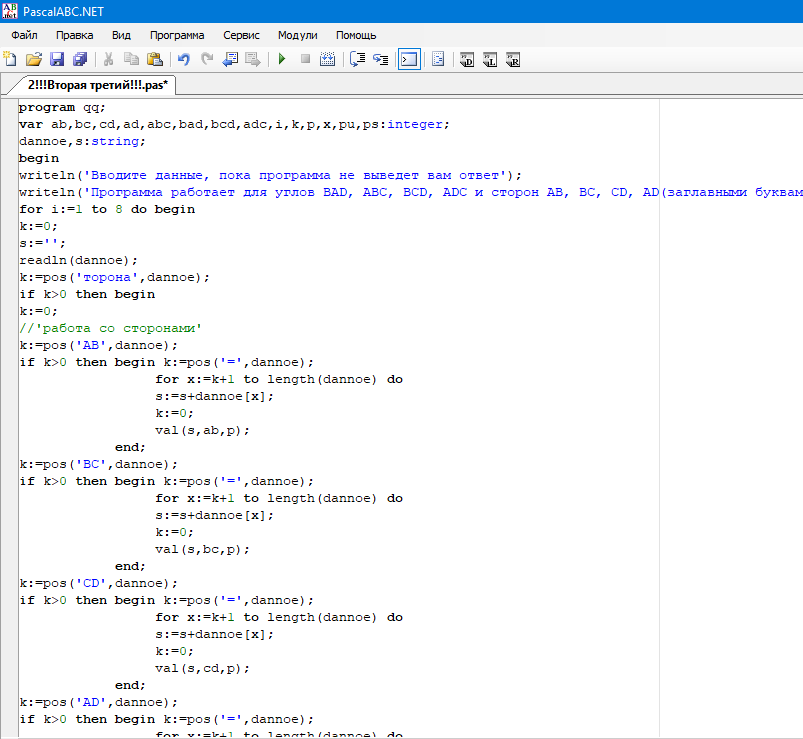
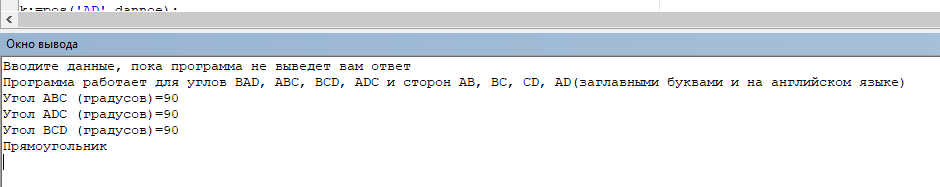
Программа работает для углов BAD, ABC, BCD, ADC и сторон AB, BC, CD, AD (заглавными буквами и на английском языке)

Угол ABC (градусов)=90

Угол ADC (градусов)=90

Угол BCD (градусов)=90

Выходные данные

Прямоугольник

**Задание 3**

Разработать программу, которая генерирует неравномерный двоичный код, допускающий однозначное декодирование.

**program** qq;

**var** n1,n2,i,max,c,k:integer;

max1,s,s1:string;

**begin**

writeln('Введите первое число: ');

read(n1);

**if** n1=0 **then else begin** writeln('Первое значение начинается с 0');**exit**;**end**;

writeln('Введите второе число: ');

read(n2);

str(n2,s);

**if** s[1]='1' **then else begin** writeln('Второе значение начинается с 1');**exit**;**end**;

write('0');

s1:=s;

**for** i:=1 **to** length(s) **do**

max1:=max1+'9';//находим предел последовательности

val(max1,max,k);

**for** i:=n2 **to** max **do begin**

str(i,s);c:=0;//ищем элементы последовательности

**for** k:=1 **to** length(s) **do begin**

**if** (s[k]='0')**or**(s[k]='1') **then** inc(c);

**if** c=length(s) **then** write(i:length(s1)+1);

**end**;

**end**;

**end**.

Входные данные

Введите первое число:

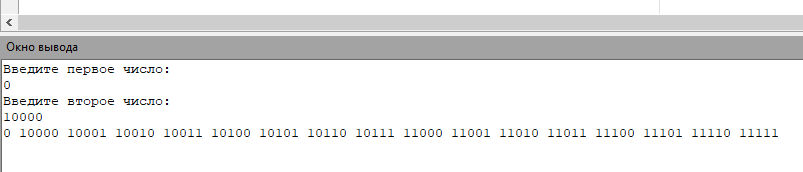
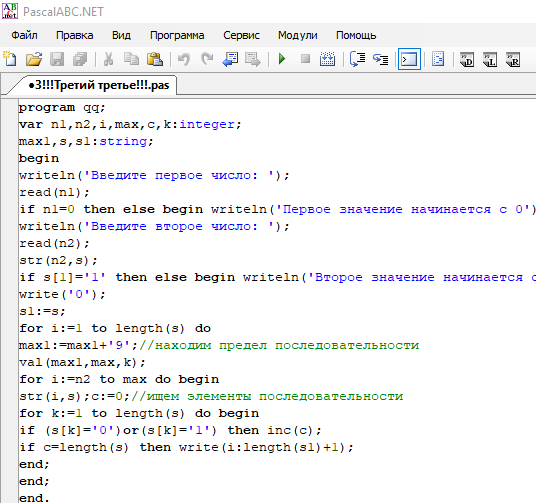
0

Введите второе число:

10000

Выходные данные

0 10000 10001 10010 10011 10100 10101 10110 10111 11000 11001 11010 11011 11100 11101 11110 11111



**Задание 4**

Разработать программу, которая по заданным IP-адресу узла и маске подсети определяет десятичные значения байтов в адресе сети (сетевом адресе).

**program** qq;

**var** l,i,j,c,k,p1,p2,p3,p4,m1,m2,m3,m4:integer;

adres,masca,ms1,ad,ma1,ma2,ma3,ma4,pi1,pi2,pi3,pi4,sa1,sa2,sa3,sa4:string;

**function** deci(n1:string):string;//функция, переводящая двоичное число в десятичное

**var** ig,yg,xg,ng,kg,kg1:integer;

sg:real;

sj:string;

**begin**

val(n1,ng,kg1);

xg:=ng;kg:=ng;

**while** xg>0 **do**

**begin**

xg:=xg **div** 10;

inc(yg);

**end**;

**for** ig:=0 **to** yg-1 **do**

**begin**

sg:=(kg **mod** 10)\*Power(2,ig)+sg;

kg:=kg **div** 10;

**end**;

str(sg,sj);

deci:=sj;

**end**;

**function** conu(ip,ms:string):string;//функция для поразрядной конъюкции,

// которая нужна для вычисления адреса сети

**var** kc,ic,ch1,ch2,k4,ch3:integer;

chl,ch:string;

**begin**

**if** length(ip)>length(ms) **then begin**

kc:=length(ip)-length(ms);

**for** ic:=1 **to** kc **do**

ms:='0'+ms;

**end**;

**if** length(ip)<length(ms) **then begin**

kc:=length(ms)-length(ip);

**for** ic:=1 **to** kc **do**

ip:='0'+ip;

**end**;

**for** ic:=1 **to** length(ip) **do begin**

val(ip[ic],ch1,k4);

val(ms[ic],ch2,k4);

ch3:=ch1\*ch2;

str(ch3,chl);

ch:=ch+chl;

**end**;

conu:=ch;

**end**;

**function** dvoi(n:integer):integer;//функция, переводящая десятичное число в двоичное

**var** a1,i1,s1,k1:integer;s:string;b1:char;

**begin**

a1:=n;

**while** a1 > 0 **do begin**

**if** a1 **mod** 2 = 0 **then**

**begin**

s := s + '0';

a1 := a1 **div** 2;

**end**;

**if** a1 **mod** 2 = 1 **then begin**

s := s + '1';

a1 := a1 **div** 2;

**end**;

**end**;

a1 := length(s) **div** 2;

**for** i1 := 1 **to** a1 **do begin**

b1 := s[i1];

s[i1] := s[length(s) - i1 + 1];

s[length(s) - i1 + 1] := b1;

**end**;

val(s,s1,k1);

dvoi:=s1;

**end**;

**begin**

write('IP-адрес узла: ');

readln(adres);

write('Маска подсети: ');

readln(masca);

adres:=adres+'.';

masca:=masca+'.';

**for** i:=1 **to** length(adres) **do begin**//применение заданных функций

**if** adres[i]='.' **then begin** inc(c);

**if** c=1 **then begin** val(ad,p1,k);ad:='';**end**;

**if** c=2 **then begin** val(ad,p2,k);ad:='';**end**;

**if** c=3 **then begin** val(ad,p3,k);ad:='';**end**;

**if** c=4 **then begin** val(ad,p4,k);ad:='';**end**;

**end else**

ad:=ad+adres[i];

**end**;

c:=0;

**for** i:=1 **to** length(masca) **do begin**

**if** masca[i]='.' **then begin** inc(c);

**if** c=1 **then begin** val(ms1,m1,k);ms1:='';**end**;

**if** c=2 **then begin** val(ms1,m2,k);ms1:='';**end**;

**if** c=3 **then begin** val(ms1,m3,k);ms1:='';**end**;

**if** c=4 **then begin** val(ms1,m4,k);ms1:='';**end**;

**end else**

ms1:=ms1+masca[i];

**end**;

m1:=dvoi(m1);m2:=dvoi(m2);m3:=dvoi(m3);m4:=dvoi(m4);

str(m1,ma1);str(m2,ma2);str(m3,ma3);str(m4,ma4);

p1:=dvoi(p1);p2:=dvoi(p2);p3:=dvoi(p3);p4:=dvoi(p4);

str(p1,pi1);str(p2,pi2);str(p3,pi3);str(p4,pi4);

sa1:=conu(ma1,pi1);sa2:=conu(ma2,pi2);sa3:=conu(ma3,pi3);sa4:=conu(ma4,pi4);

write('Сетевой адрес: ');

writeln(deci(sa1),'.',deci(sa2),'.',deci(sa3),'.',deci(sa4));

**end**.

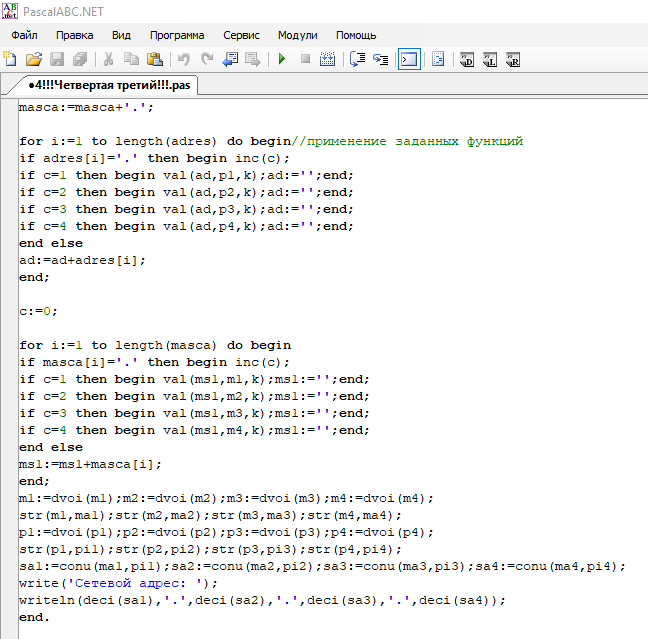
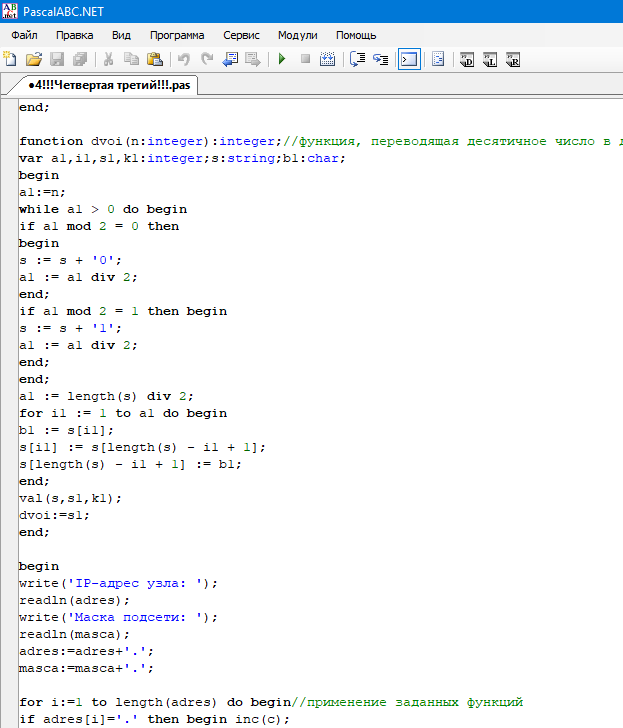
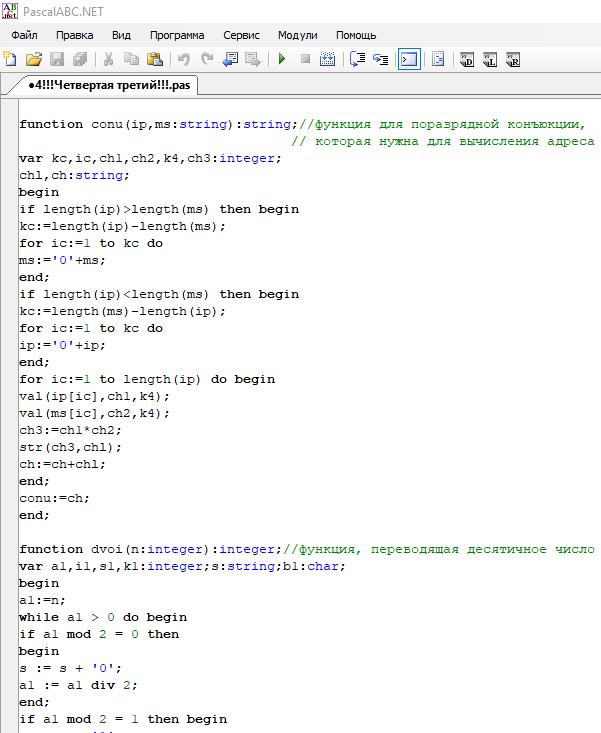
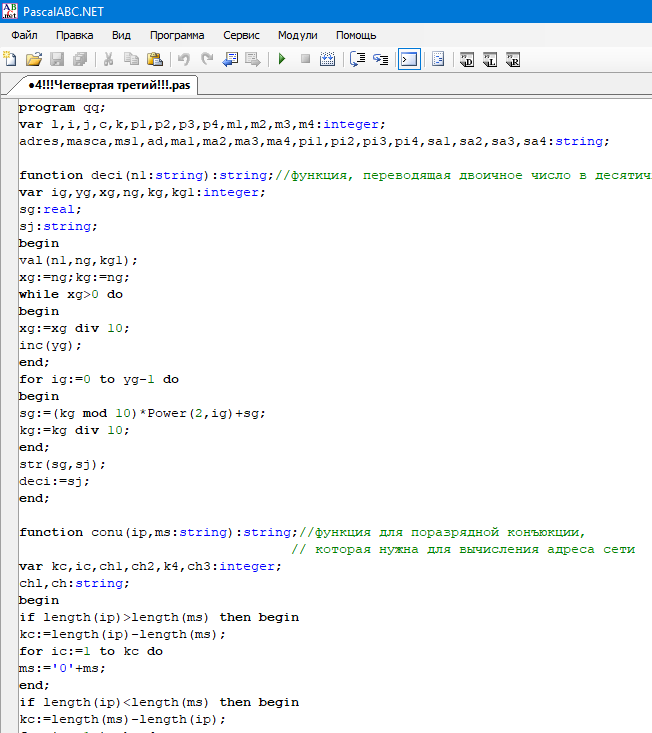
Входные данные

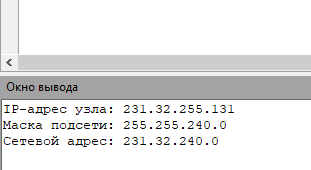
IP-адрес узла: 231.32.255.131

Маска подсети: 255.255.240.0

Выходные данные

Сетевой адрес: 231.32.240.0





**Задание 5**

Разработать программу, которая определяет информационный объем всех результатов наблюдений влажности воздуха на метеорологической станции в байтах для некоторого фиксированного количества измерений.

**program** qq;

**var** n:integer;

**begin**

{минимальное приближенное число к 100- это 128 (2^7),

т.е. результат одного измерения возможно внести лишь с помощью 7 бит ►N\*7.

Что бы перевести биты в байты, нужно умножить на 8, т.е. N\*7/8}

write('Количество измерений: ');

readln(n);

writeln('Информационный объем результатов наблюдений в байтах: ',n\*7/8);

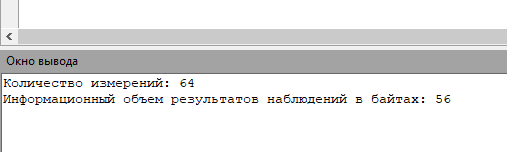
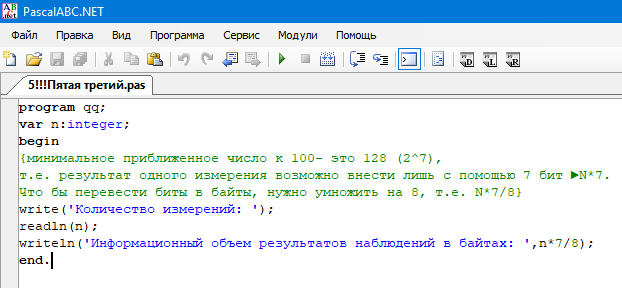
**end**.

Входные данные

Количество измерений: 64

Выходные данные

Информационный объем результатов наблюдений в байтах: 56



**Задание 6**

Разработать программу, которая подсчитывает количество нуклеотидов для заданной в программе строки.

**program** qq;

**var** a,c,g,t,i:integer;

s:string;

**begin**

write('Введите последовательность нуклеотидов ');

readln(s);

**for** i:=1 **to** length(s) **do begin**//последовательно считаем кол-во каждого символа

**if** (s[i]='A') **or** (s[i]='a') **then** inc(a);

**if** (s[i]='C') **or** (s[i]='c')**then** inc(c);

**if** (s[i]='G') **or** (s[i]='g')**then** inc(g);

**if** (s[i]='T') **or** (s[i]='t')**then** inc(t);

**end**;

writeln('Количество нуклеотидов A | C | G | T: ');

writeln(' '\*23,a,' ',c,' ',g,' ',t);

**end**.

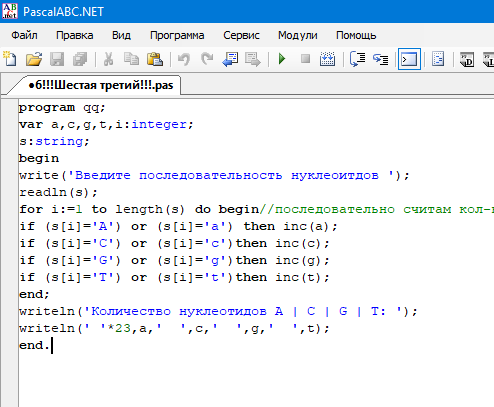
Входные данные

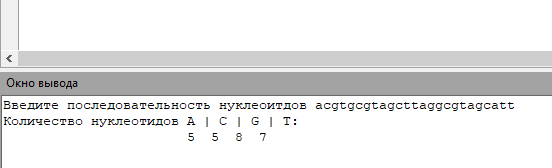
Введите последовательность нуклеотидов acgtgcgtagcttaggcgtagcatt

Выходные данные

Количество нуклеотидов A | C | G | T:

5 5 8 7





**Задание 7**

Разработать программу, которая выполняет построение комплементарной цепочки ДНК для заданной строки

**program** qq;

**var** s,s1:string;

i,k:integer;

**begin**

readln(s1);

**for** i:=length(s1) **downto** 1 **do**

s:=s+s1[i];//переворачиваем цепочку

s:=s+' ';//присвоил лишний символ, чтобы программа не остановилась на предпоследнем символе

i:=1;

**while** i<>length(s) **do begin**

**if** s[i]='A' **then begin** s[i]:='T';i:=i+1;**end**;//Заменяет аденин на тимин

**if** s[i]='T' **then begin** s[i]:='A';i:=i+1;**end**;//Заменяет тимин на аденин

**if** s[i]='G' **then begin** s[i]:='C';i:=i+1;**end**;//Заменяет гуанин на цитозин

**if** s[i]='C' **then begin** s[i]:='G';i:=i+1;**end**;//Заменяет цитозин на гуанин

**if** s[i]=' ' **then break**;

**end**;

writeln(s);

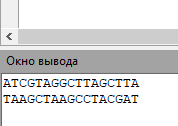
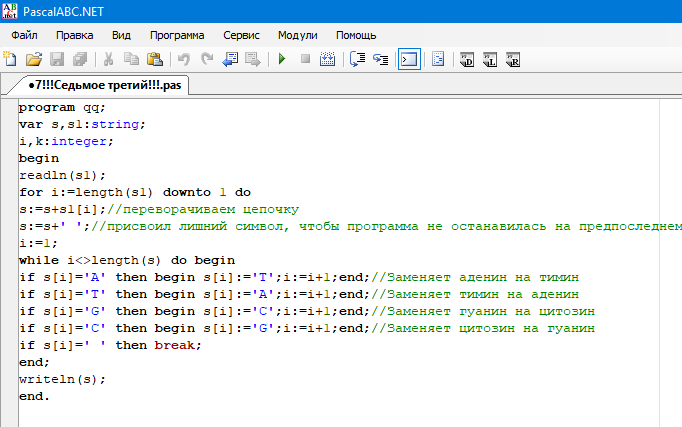
**end**.

Входные данные

ATCGTAGGCTTAGCTTA

Выходные данные

TAAGCTAAGCCTACGAT



**Задание 8**

Разработать программу, которая выполняет подсчет количества точечных мутаций.

**program** qq;

**var** s,s1:string;

i,c:integer;

**begin**

write('Введите первую цепочку ');

readln(s);

write('Введите вторую цепочку ');

readln(s1);

**if** s>=s1 **then begin**//уравниваем кол-во символов в обеих строчках,

//затем ищем разные символы на одинаковых местах

c:=length(s)-length(s1);

**if** s=s1 **then else for** i:=1 **to** c **do** s1:=s1+'\_';

**for** i:=1 **to** length(s) **do begin**

**if** s1[i]='\_' **then break**;

**if** s[i]=s1[i] **then else** inc(c);**end**;

**end else begin**

c:=length(s)-length(s1);

**if** s=s1 **then else for** i:=1 **to** c **do** s:=s+'\_';

**for** i:=1 **to** length(s1) **do begin**

**if** s[i]='\_' **then break**;

**if** s[i]=s1[i] **then else** inc(c);**end**;

**end**;

writeln('Расстояние Хэмминга: ',c);

**end**.

Входные данные

Введите первую цепочку GAGCCTACTAACGGGATCATCGTAATGACGGCCT

Введите вторую цепочку CATCGTAATGACGGCCTCATCGTAATGACGGCCT

Выходные данные

Расстояние Хэмминга: 7

