

Нелінійні алгоритми мовою Pascal

Автор: Негода Сергій Петрович

□

Практична робота 11.

Алгоритми розгалуження.

Задача 1. Василько та Юлія грають в таку гру. Спочатку кожен записує на папері свій прогноз – число від 1 до 6. Потім вони кидають гральний кубик з числами від 1 до 6 на гранях. Чий прогноз виявляється ближчим до того числа, що випало, той і переміг. Треба написати програму для визначення переможця.

Технічні умови. Програма Forga читає з пристрою стандартного введення три числа через пропуски (пробіли) – прогноз Василька, Юлі та результат кидання кубика. Програма виводить "V", якщо переміг Василько, "J" якщо Юлія або "D" – якщо прогноз обох однаково близький до результату (тобто переможця виявити неможливо).

Програма мовою Pascal

```

programfora;                                {оголошення назви алгоритму}

vara,b,c : integer;                       {оголошення змінних цілих величин a,b,c}

begin writeln;                            {оголошення початку роботи алгоритму}

readln(a); readln(b); readln(c);          {оголошення про введення величин a, b, c}

    if ( (abs(c-a)) > (abs(c-b)) ) then write('j'); {якщо |c-a| > |c-b|, тоді написати j}

if ( (abs(c-a)) < (abs(c-b)) ) then write('v'); {якщо |c-a| < |c-b|, тоді написати v}

if ( (abs(c-a)) = (abs(c-b)) ) then write('d'); {якщо |c-a| = |c-b|, тоді написати d}

end.

```

Протестувати алгоритм на правильність на прикладах:

1) Введення 3 4 5 Виведення J; 2) Введення: 1 6 2 Виведення V; 3) Введення: 4 4 3 Виведення D; 4) Введення 6 5 2 Виведення J; 5) Введення: 1 6 2 Виведення V.

Завдання 3. Відомо, що книжкова полиця вміщає однакових товстих книг, але $k+1$ -а книга вже не влазить. Так само на неї можна поставити однакових тонких книг, $m+1$ -а вже не влізе. Скласти алгоритм, який знаходить можливість, щоб на полиці помістилися одночасно: товстих та тонких книг.

Розв'язання. Якщо числовий вираз $n/k + p/m \leq 1$, то можна, якщо числовий $n/k + p/m > 1$, то не можна помістити одночасно книги на полицю.

Алгоритм мовою Pascal (використовує повне розгалуження, «якщо-то, інакше»)

```

program BIBLIO; var k,m,n,p,h:real; begin
writeln('введіть число товстих книг  $2 < k < 10^9$  k='); readln(k);
writeln('введіть число тонких книг  $2 < m < 10^9$  m='); readln(m);
writeln('введіть даних товстих книг  $2 < n < 10^9$  n='); readln(n);
writeln('введіть даних тонких книг  $2 < p < 10^9$  p='); readln(p); h:=(n/k)+(p/m);
if (h<1) or (h=1) thenwrite(' книги можна помістити') {розгалуження для результату}
else write('не можна помістити книги'); {інакше то вивід результату не можна}writeln
('h=', h); end.

```

Протестувати алгоритм на правильність на прикладах:

1)Введення 3 4 5 6 Виведення ? 2) Введення: 1 6 2 8 Виведення ?; 3) Введення: 4 4 3 9 Виведення ?; 4)Введення 6 5 2 10 Виведення ?; 5) Введення: 2 6 7 9 Виведення ?

Завдання 4.Самостійно скласти і реалізувати алгоритм впорядкування виразів: $(n/k)+(p/m)-(m/p)-(k/m)$; та $(p/k)+(n/m)-(k/n)-(n/p)$ в порядку зростання для дробових чисел k,m,n,p .

Протестувати алгоритм на правильність на прикладах:

1)Введення 3 4 5 6 Виведення ? 2) Введення: 1 6 2 8 Виведення ?; 3) Введення: 4 4 3 9 Виведення ?; 4)Введення 6 5 2 10 Виведення ?; 5) Введення: 2 6 7 9 Виведення ?

Завдання 5.Самостійно скласти і реалізувати алгоритм впорядкування виразів: $(1/k)+(1/m)-1/p)-(1/n)$; та $(1/k)+(1/p)-(1/m)-(1/n)$ в порядку зростання для дробових чисел k,m,n,p .

Протестувати алгоритм на правильність на прикладах:

1)Введення 3 4 5 6 Виведення ? 2) Введення: 1 6 2 8 Виведення ?; 3) Введення: 4 4 3 9 Виведення ?; 4)Введення 6 5 2 10 Виведення ?; 5) Введення: 2 6 7 9 Виведення ?

Практична робота 12.

Програмування циклів мовою Pascal.

Завдання 1. Скласти і реалізувати алгоритм підрахунку різних букв у слові мовою Pascal.

```

program Wordlitera; { оголошення назви алгоритму }
var s:string; r:real; i,j,n:integer; {оголошення змінних величин алгоритму:
рядкові, дійсні, цілі}

```

```

begin                { оголошення початку алгоритму }

r:=0;                { оголошення присвоєння нуля дійсній змінній }

readln(s);           { оголошення про зчитування з екрану рядкової змінної }

for i:=1 to length(s) do begin  { оголошення циклу з лічильником i в алгоритмі }

    n:=0;                { оголошення про присвоєння нуля }

    for j:=1 to length(s) do begin  { оголошення циклу з лічильником j в алгоритмі }
}

    ifs[i]=s[j] then inc(n);  { оголошення про перевірку на однаковість букв }

    end;                { оголошення про кінець циклу }

    r:=r+1/n;           { оголошення лічильника кількості різних букв }

end; writeln('кількість різних букв у слові r = ', r:1:0); end.  { оголошення про кінець циклу, алгоритму }

```

Протестувати правильність виконання алгоритму: Ввести: **карта; агов; математика; інформатика-а-а.**

Завдання 2. Скласти і реалізувати алгоритм знаходження цілих дільників натурального числа мовою Pascal.

```

program Numer;                { оголошення назви алгоритму }

var a,n,c,d: integer;  { оголошення цілих величин a,n,c,d в алгоритмі }

begin                            { початок алгоритму }

    readln(a);  { оголошення про зчитування з екрану змінної }

    n:=1;      { оголошення про присвоєння лічильнику 1 }

    while ( n <= sqrt(a) ) do begin  { оголошення циклу з передумовою(допоки ... виконати...) в алгоритмі }

        c:=a mod n;  { оголошення про знаходження остачі від ділення a:n }

        d:=a div n;  { оголошення про знаходження цілої частини від ділення a:n }

        if c = 0 then begin writeln(n);  {Перевірка подільності націло, якщо остача =0, то написати дільник }

            if n <> d then writeln( d );  {Перевірка неоднаковості дільників, якщо дільники різні, то написати дільник }

        end;    inc(n); end; end.  { оголошення про кінець циклу та алгоритму }

```

Протестувати правильність виконання алгоритму: Ввести чисел: **4; 6; 8; 94; 96; 80; 99; 100; 272; 558; 997**

Завдання 3. Скласти і реалізувати алгоритм знаходження простих натуральних чисел мовою Pascal.

Program Prime;

```

const LIMIT = 50;           {оголошення постійної величини в алгоритмі: цілі}
var i,j,lim: integer;      {оголошення змінних величин i, j, lim в алгоритмі: цілі}
begin                       {початок основної програми алгоритму}

  writeln;                 {початок з нового рядку}

  for i:=1 to LIMIT do begin {оголошення про початок циклу з лічильником i}
    j:=2; lim:=round(sqrt(i)); {оголошення про присвоєвоння лічильнику 2}
    while (i mod j <> 0) and (j <= lim) do inc(j); {цикл з складеною передумовою}
    if (j > lim) then write( i, ' '); end; end. { перевірка, вивід і оголошення про кінець}

```

Протестувати правильність виконання алгоритму: Замінити: LIMIT = 50 на LIMIT = 100; LIMIT = 997.

Завдання 4. Скласти і реалізувати алгоритм знаходження суми цифр натурального числа мовою Pascal.

Program Summa;

```

var a,x, i,s: integer;

begin

  writeln('введіть ціле число'); readln(a);

  x:=a; s:=0;

  while ( x < > 0 ) do begin

    s := s + (x mod 10);

    x := x div 10;    end;

  writeln( 'Сума цифр числа ', a, ' s= ', s); end.

```

Протестувати правильність виконання алгоритму: Ввести чисел: **4; 6; 8; 94; 96; 80; 99; 100; 272; 558; 997**

Завдання 5. Самостійно скласти і реалізувати алгоритм знаходження послідовності чисел,

що утворюється сумою усіх цифр натурального числа та додаванням цієї суми до самого числа мовою Pascal.

Наприклад: $A(15) = 1 + 5 + 15 = 26$, $A(20) = 2 + 0 + 20 = 22$, $A(95) = 9 + 5 + 95 = 109$, $A(24) = 2 + 4 + 24 = 30$, $A(111) = 1 + 1 + 1 + 111 = 114$.

Завдання 6. Самостійно скласти і реалізувати алгоритм знаходження послідовності чисел, що утворюється добутком усіх цифр натурального числа та множенням цього добутку на саме число мовою Pascal.

Наприклад: $A(15) = 1 * 5 * 15 = 75$, $A(20) = 2 * 0 * 20 = 0$, $A(95) = 9 * 5 * 95 = 4275$, $A(24) = 2 * 4 * 24 = 192$, $A(111) = 1 * 1 * 111 = 111$.

Практична робота 13.

Цикли і розгалуження

Завдання 1. Скласти і реалізувати алгоритм мовою Паскаль, який перевіряє приналежність натурального числа до чисел Фібоначчі, тобто приналежність до ряду чисел, в якому кожне наступне число дорівнює сумі двох попередніх чисел (наприклад: $1 + 1 = 2$; $1 + 2 = 3$; $2 + 3 = 5$; $3 + 5 = 8$ і т.д.). До ряду чисел Фібоначчі належать: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040,

Програма мовою Pascal.

```

program Fibonacci;    {Оголошення назви алгоритму}

var   a,b,c,n: integer;  asc: boolean; {Оголошення змінних величин : цілих та :
логічних}

begin   write('Введіть натуральне число N='); readln(n); {Оголошення про числа }

   a:=0;b:=1;c:=0;           {Оголошення початкових значень для цілих змінних}

   asc:=false;              {Оголошення початкового значення для логічної змінної}

   while c<n do begin           {Оголошення початку циклу з передумовою}

       c:=a+b; a:=b;   b:=c;    { Сума двох попередніх змінних і переприсвоєння результатів
}

       if n=c then asc:=true; b:=c;   end; { Перевірка критерію числа Фібоначчі і кінець
циклу}

       if asc = true then writeln(n,' - це число Фібоначчі') {Перевірка і вивід числа
Фібоначчі}

       else writeln(n,' - це не є числом Фібоначчі'); end.      {Перевірка і вивід

```

результату}

Протестуйте правильність роботи цього алгоритму. **Введення: 7; 8; 9; 54; 55; 832040.**

Завдання 2. Скласти і реалізувати алгоритм мовою Паскаль, який перевіряє, чи декілька натуральних чисел являються взаємно простими. Два числа взаємно прості, якщо у них найбільший спільний дільник дорівнює 1. Наприклад: 2 та 3 – це взаємно прості числа; 4 та 8 – це не взаємно прості числа. Програма мовою Pascal.

```

program zysla2;                                {Оголошення назви алгоритму}

var a: array[1..100] of integer;              {Оголошення про ряд чисел(числовий масив
1x100) в алгоритмі}

i,max,n,j: integer;                          {Оголошення про цілі змінні в алгоритмі}

begin   write('vvedite koli4estvo 4isel:'); readln(n); {Оголошення про введення кількості
чисел}

for i:=1 to n do                               {Оголошення початку циклу з лічильником}

begin   write(i, ' 4islo:'); readln(a[i]); end;      {Оголошення введення даних
чисел по порядку}

max:=a[1];

for i:=2 to n do if max<a[i] then max:=a[i];      {Оголошення початку циклу з
лічильником і перевірка}

for i:=2 to max do                              {Оголошення початку циклу з
лічильником}

for j:=1 to n do                                 {Оголошення початку циклу з
лічильником}

if a[j] mod i=0 then begin write('ТАК');        {Оголошення перевірки подільності
чисел з лічильником}

readln; end

else begin write('НІ'); readln; end;          {Оголошення кінця перевірки та циклу з
лічильником}

readln; end.

```

Протестуйте правильність роботи алгоритму. **Введення: {2; 8; 9}=ТАК, {2; 25; 10}=НІ, {3; 7; 14; 35}=НІ**

Завдання 3. Скласти і реалізувати алгоритм мовою Паскаль, який перевіряє ряд випадкових натуральних чисел, які при діленні на сім дають остачу 2 та 1 і виводить такі числа на екран.

Програма мовою Pascal

Program Ostaci;

{Оголошення назви алгоритму}

Const n = 5; {Оголошення про постійну величину: **n** - це кількість чисел в ряді}**var x : array [1..n] of integer; i : integer;** {оголошення про ряд чисел(числовий масив) в алгоритмі}**begin****for i := 1 to n do x[i] := random(100);** {цикл формування ряду випадкових чисел, менше 100} **for i := 1 to n do write(x[i]:3); writeln;** {оголошення про виведення на екран ряду випадк. чисел}**for i := 1 to n do** {цикл з лічильником для перевірки ряду випадкових чисел на остачі}**if (x[i] mod 7 = 2) or (x[i] mod 7 = 1)** {оголошення про перевірку чисел на остачі на 2 та 1 }**then write(x[i]:3, ' (, i, ') ');** {оголошення про виведення на екран ряду перевірених чисел}**writeln; readln; end.**

Протестуйте правильність роботи алгоритму. 1) **Введення: Const n = 7; Const n = 19; Const n = 22;** 2) **Змінити у програмі перевірку остач:** перевірити числа на остачі: 4 та 6, перевірити на остачі: 0 та 6. 3) **Змінити у програмі перевірку подільності на 9:** перевірити числа на остачі: 1 та 3, перевірити на остачі: 0 та 5.

Практична робота 14.

Програмування циклів мовою Pascal.

Завдання 1. Ліцей проводить вибори до учнівського парламенту. Скількома способами можна обрати цей парламент, якщо до парламенту подано: k заявок від 8-класників, m заявок від 9-класників, n заявок від 10-класників. Скласти і реалізувати алгоритм підрахунку усіх способів утворення шкільного парламенту мовою Pascal, за умови, що серед вибраних є не менше одного представника із трьох паралелей, і до парламенту проходить тільки не більше половини від ТИХ, ХТО подав заявки. А саме, у парламенті має зайти не більше: $0,5k+1$ осіб від 8-класників, не більше $0,5m+1$ осіб від 9-класників, не більше $0,5n+1$ осіб від 10-класників.

Розв'язання. У парламент може зайти: або один, або два, або три, ..., або $0,5k+1$ від 8-класників. Кількість способів, обчислюємо як сума комбінацій: $C_{k/2+1}^1 + C_{k/2+1}^2 + C_{k/2+1}^3 + \dots + C_{k/2+1}^{k/2}$. Аналогічно знаходимо кількість способів для 9-класників: $C_{m/2+1}^1 + C_{m/2+1}^2 + C_{m/2+1}^3 + \dots + C_{m/2+1}^{m/2}$. Аналогічно знаходимо кількість способів для 10-класників: $C_{n/2+1}^1 + C_{n/2+1}^2 + C_{n/2+1}^3 + \dots + C_{n/2+1}^{n/2}$. А кількість вибору того, що у парламенті мають бути

представники від трьох паралелей:

$$P_1 = (C_{k/2+1}^1 + C_{k/2+1}^2 + C_{k/2+1}^3 + \dots + C_{k/2+1}^{k/2}) (C_{k/2+1}^1 + C_{k/2+1}^2 + C_{k/2+1}^3 + \dots + C_{k/2+1}^{k/2})$$

program Parlament;

var i, k, m, n, v1, d1, b1, v2, d2, b2, v3, d3, b3:integer;

begin

writeln('Введіть число заявок від 8-класників: k= '); readln(k); k:=(k div 2)+1;

writeln('Введіть число заявок від 9-класників: m= '); readln(m); m:=(m div 2)+1;

writeln('Введіть число заявок від 10-класників: n= '); readln(n); n:=(n div 2)+1;

v1:=1; d1:=1; b1:=1; v2:=1; d2:=1; b2:=1; v3:=0; d3:=0; b3:=0; {початкові значення змінних}

for i:=1 to k do begin v1:=i*v1; v2:=(k-i+1)*v2; v3:=(v2 div v1)+v3; end; {цикл 8-их класів}

for i:=1 to m do begin d1:=i*d1; d2:=(m-i+1)*d2; d3:=(d2 div d1)+d3; end; {цикл 9-их класів}

for i:=1 to n do begin b1:=i*b1; b2:=(n-i+1)*b2; b3:=(b2 div b1)+b3; end; {цикл 10-их класів}

writeln('Кількість способів обрати членів парламенту від окремих паралелей:');

writeln(' v3= ', v3, ' способів тільки від 8-класників; ');

writeln(' d3= ',d3, ' способів тільки від 9-класників;');

writeln(' b3= ',b3, ' способів тільки від 10-класників; ');

writeln('Кількість способів обрати новий учнівський парламент ліцею:');

writeln(' p= ', v3*d3*b3, ' способів'); writeln(' '); end.

Протестуйте правильність роботи алгоритму: {k;m;n}={ (1;1;1)=1; (2;2;2)=27; (3;3;3)=27; (4;4;4)=343; (3;4;2)=63 }

Практична робота 15.

Алгоритми різних типів

Завдання 1. Створити, реалізувати алгоритм мовою Pascal, який знаходить значення функції. Це лінійний алгоритм, бо містить тільки вираз з алгебраїчними діями.

```
program tabul1;
```

```
var x, y, z, b : real;
```

```
begin write('введіть x :'); readln(x); write('введіть y :'); readln(y);
```

```
write('введіть z :'); readln(z);
```

```
b:=sqr(sin(x))+(sqr(y)+sin(sqr(z))/cos(sqrt(z)))/(z+exp(-x))+abs(55-2*random(99));
```

```
writeln('значення b=', 5*b-3);end.
```

Протестуйте цей алгоритмякщо: 1) x= 10; y=7; z=9. 2) x= 100; y=25; z=49.

Завдання 2. Створити, реалізувати алгоритм мовою Pascal, який містить повне розгалуження **if .. else** і обчислює значення декількох виразів при виконанні умов на знак числа x. Це нелінійний алгоритм, бо має розгалуження на дві умови «+x» та «-x».

```
program tabul2;
```

```
var k,a, r,l, n,q,x: integer;
```

```
begin
```

```
x:=48-random(80); writeln('початкове значення x :=',x);
```

```
n:=random(90); writeln('початкове значення n :=',n);
```

```
a :=random(70); writeln('початкове значення a :=',a);
```

```
q:=random(60); writeln('початкове значення q :=',q);
```

```
l:=random(60); writeln('початкове значення l :=',l);
```

```
k:=random(60); writeln('початкове значення k :=',k);
```

```
if x>0 then begin k:=2*k+ k*k+25; r:=2*a+a*a+24; l:=0; q:=0; end
```

```
else begin l:=2*l+1; q:=q*a+3;k:=0; r:=0; end;
```

```
writeln('число x :=', x); writeln('число k:=', k); writeln('число r:=', r);
```

```
writeln('число a :=', a); writeln('число q:=', q); writeln('число l :=',l); end.
```

Завдання 3. Створити, реалізувати алгоритм мовою Pascal, який містить вибір **case..** Це нелінійний алгоритм, бо містить вибір на 6 випадків для цілих значень k від 0 до 5.

```
program tabul3;
```

```
var a, r, l, n, y, x: real; k: integer;
```

```
begin
```

```
  x:=random(80);  writeln('початкове значення x :=',x);
```

```
  n:=random(90);  writeln('початкове значення n :=',n);
```

```
  a :=random(70);  writeln('початкове значення a :=',a);
```

```
  y:=random(60);  writeln('початкове значення y :=',y);
```

```
  l:=random(60);  writeln('початкове значення l :=',l);
```

```
  k:=random(6);  writeln('початкове значення k :=',k);
```

```
case k of
```

```
  1: a:=abs(20*sqrt(a) - 40.567/ln(a));
```

```
  2: n:=25/sqr(l)+6.89*exp(-n);
```

```
  3: x:=6/sin(-n)+7*cos(y);
```

```
  4: r:=6*sin(n)+7.7*cos(-y);
```

```
  else y:=0; n:=0; x:=0; r:=0; a:=0; end;
```

```
writeln('r:=', r); writeln('y :=', y); writeln('k:=', k);  writeln('n:=', n);
```

```
writeln('числоa :=', a); writeln('числоl:=', l); writeln('*****'); end.
```

Завдання 4. Створити, реалізувати алгоритм мовою Pascal, який містить повторення з лічильником k. Це нелінійний алгоритм, бо має два цикли з лічильниками.

```
program tabul4;
```

```
var x : array[1..5] of real;      a,b,y : real;      i : integer;
```

```
begin  a:=round(sqrt(random(45)))-2*sqr(round(random(70)))+25;  writeln(' a:=',a);
```

```
  b :=10*round(40/sqr(a)) -3*round(random(70));  writeln('початкове значення b :=',  
b);
```

```
  for i:=1 to 5 do begin x[i]:=10- round(random(70)); write('  x[',i,']=', 2+6*x[i]);  
end;
```

```
  for i:=1 to 5 do begin y:=-a*sqr(x[i])+b; writeln('  x=',8*x[i], '  y=', 30+0.34*y); end;  
end.
```

Практична робота 16.

Економічні алгоритми на рентабельність

Завдання 1. Вартість автомобіля дорівнює A грн, а його капітальний ремонт вимагає r грн. Встановлено, що автомобіль може працювати без ремонту n років, а з ремонтом m років. За яких умов між A , r , n , m витрати ремонту являються рентабельними? При цьому треба врахувати, що після ремонту потужність автомобіля дорівнює потужності нового автомобіля.

Розв'язання. A/n - це середня вартість місячної експлуатації нового автомобіля. $A+r$ - це сумарна вартість автомобіля і ремонту. $(A+r)/m$ - це середня вартість річної експлуатації автомобіля після ремонту. Капітальний ремонт автомобіля буде рентабельним, тобто окупить себе, тільки в тому випадку, якщо середня вартість експлуатації автомобіля після ремонту буде більше ніж середня вартість експлуатації до ремонту. Тому отримуємо нестрогу нерівність, $(A+r)/m \leq A/n$, звідки отримуємо

$$r \leq A(m-n)/n.$$

Створити та реалізувати мовою Паскаль алгоритм, який обчислює рентабельність ремонту автомобіля при заданих A , r , n , m . Цей алгоритм використовує вкладені цикли і розгалуження в повній формі.

programremont1;

var a, r, m, n, b, i: integer; {оголошуються змінні типу: цілі}

begin writeln(' Вводяться початкові випадкові числа A, r, n, m. ');

a:=(10000+random(100000)); write('якщо вартість автомобіля: a =', a); writeln;

r:=(1000+random(90000)); write('якщо вартість ремонту: r =', r); writeln;

n:= 1+random(12); write('якщо кількість місяців без ремонту: n=', n); writeln;

m:= 10+random(18); write('якщо кількість місяців з ремонтом: m =', m); writeln;

for i:=1 to n do begin {два цикли з лічильником для обчислення рентабельності}

for i:=1 to m do begin b:=a*(m-n) div n;

if (r<b) or (r=b) then { повне розгалуження для перевірки рентабельності }

writeln(' Варто ремонтувати автомобіль, бо дешевий кап.ремонт ', r, ' =<', b)

else writeln(' Не варто ремонтувати автомобіль, бо дорогий кап.ремонт ', r, ' >', b);

writeln; end; end; end.

Протестуйте цей алгоритм декілька разів і порівняйте результати обчислень на кожному кроці. Спробуйте самостійно змінити в алгоритмі діапазон випадкових чисел.

Завдання 2. Затрати при перевезенні вантажів двома видами транспорту обчислюється за двома лінійними формулами: $y_1 = a + bx$, $y_2 = m + kx$, де x - це відстань перевезення в сотнях кілометрів, y грн. - це транспортні витрати при перевезенні вантажу також $a > 0$, $b > 0$, $m > 0$, $n > 0$. Знайти, на які відстані і яким видом транспорту перевезення вантажів буде рентабельним?

Розв'язання. Дві формули $y_1 = a + bx$, $y_2 = m + kx$ задають прямі лінії в прямокутній системі координат. Одна з цих прямих лежить вище ніж друга. Ті точки, що лежать на прямій, що розміщена нижче, означають, що затрат буде менше. Є випадок, коли ці прямі перетинаються (це однакові затрати). Координати точки перетину знаходиться із системи двох рівнянь, тобто $x = (m - a) / (b - k)$ сотень км; $y = a + b * (m - a) / (b - k)$ грн. Отже, якщо перевозити на відстань менше ніж $(m - a) / (b - k)$ км, то варто користуватися першим видом транспорту, а якщо перевозити на відстань більшу ніж $(m - a) / (b - k)$ км, то варто користуватися другим видом транспорту.

Створити та реалізувати мовою Паскаль алгоритм, який обчислює і порівнює рентабельність кожного виду транспорту для деякої відстані перевезення вантажу.

program perevoz2;

var a, k, m, x, b, y, c, v1, v2: integer; {оголошуються змінні типу: цілі}

begin writeln(' Вводяться початкові випадкові числа для: $a > 0$, $b > 0$, $m > 0$, $n > 0$ ');

a := (300 + random(900)); write('якщо вартість 1-го перевізника: a =', a); writeln;

b := (29 + random(5)); write('якщо вартість палива на 100 км: b =', b); writeln;

m := (400 + random(900)); write('якщо вартість 2-го перевізника: m =', m); writeln;

k := (26 + random(4)); write('якщо вартість палива на 100 км: k =', k); writeln;

x := (1 + random(150)); write('якщо відстань перевезення вантажу: x =', x); writeln;

c := (m - a) div (b - k); write('якщо відстань перевезення вантажу: c =', c, ' то можна на 2-ох видах транспорту'); writeln;

v1 := a + b * x; write('вартість перевезення 1-им видом транспорту: v1 =', v1); writeln;

v2 := m + k * x; write('вартість перевезення 2-им видом транспорту: v2 =', v2); writeln;

if (v1 < v2) then { повне розгалуження для перевірки рентабельності }

writeln('Варто везти 1-им видом транспорту, бо бо менша вартість 1-ого ', v1, ' <', v2)

```
else writeln('Варто взяти 2-им видом транспорту, бо менша вартість 2-ого ', v1, ' >', v2);
```

```
writeln; end.
```

Протестуйте цей алгоритм декілька разів і порівняйте результати обчислень на кожному кроці. Спробуйте самостійно змінити в алгоритмі діапазон випадкових чисел.

Практична робота 17.

Нелінійні алгоритми з практичним змістом

Завдання 1. На вокзалі з потягу зійшли два пасажира і направились одночасно в один і той же пункт А. Перший пасажир половину часу йшов зі швидкістю v_1 м/год, а другу половину часу йшов зі швидкістю v_2 м/год. Другий пасажир йшов першу половину шляху зі швидкістю v_2 м/год, а другу половину шляху зі швидкістю v_1 м/год. Допоможіть слідчому, дізнатися, яку відстань долали пасажири від виходу із потяга до пункту призначення і хто першим прибуває в пункт А. Скласти і реалізувати алгоритм для вияснення, хто першим прибуває у пункт призначення і на скільки раніше, ніж інший, якщо відомо, що перший пасажир витрачає на весь свій шлях t хвилин?

```
program TOURIST_1;      {назва алгоритму}

var v1,v2,t1,t2, s1 : real;  {оголошення змінних величин: v1,v2,t1,t2, s1,s2 - це дійсні числа}

begin                    {початок виконання алгоритму}

  writeln( 'v1='); readln(v1); writeln( 'v2='); readln(v2); writeln( 't1='); readln(t1);

  s1:=(v1+v2)*t1*0.5;      {обчислення за фор-лою відстані від потяга до пункту}

  writeln('Довжина шляху пасажирів', s1, ' метрів ');      {виведення результату}

  t2:=(v1+v2)* (v1+v2)*t1*0.25/( v1*v2);  {обчислення за фор-лою часу другого пасажира}

  writeln('Час руху другого пасажира: ',t2, ' хвилин ');  {виведення результату}

  if t2 -t1=0 thenwriteln('У пункт А пасажири прибувають одночасно');

  if t2 -t1>0 thenwriteln('У пункт А раніше прибуває перший пасажирна: ', t2 -t1, ' хв. '); end.
```

Протестуйте алгоритм для таких значень $\{v_1; v_2; t_1\}$: а) (90; 80; 30)=0,1 хв; б) (85; 85; 40)=0 хв; в) (60; 80; 20)=0,42 хвилини.

Завдання 2. Якщо через рівні проміжки часу на депозитну картку вноситься деяка постійна сума K грн(періодичні внески) під складні відсотки $P\%$, то заощадження обчислюється за формулою

$$S=K*(1+p/100)*(exp(ln(1+p/100)*n)-1)/((1+p/100)-1).$$

Скласти і реалізувати алгоритм для вивчення сум грошей на депозитній картці через декілька років.

```

program BABLO_2;           {назва алгоритму}

var k,p,r, s: real;    n, i: integer; {оголошення змінних величин: k,p,r,n, s- це дійсні числа}

begin                   { початок виконання алгоритму}

k:=1000+random(100); writeln( ' Якщо сума, що вноситься на депозит k=', k);
writeln;

p:=7+random(10); writeln( ' Якщо відсоткова ставка для цього депозиту p=', p);
writeln;

n:=1+random(10); writeln( ' Якщо кількість років існування депозиту n=', n);
writeln;

s:=1+p/100; u:=1;       {обчислення початкового відсотку для виконання алгоритму}

fori:=1 ton dobegin    {виконання циклу з лічильником по рокам для обчислення грошей}

u:=u*(1+p/100); s:=1+p/100; s:= K*s*(u-1)/(s-1);

writeln( 'Кількість грошей на депозиті через', i, 'років S=', s); writeln;end;
writeln('*****'); end.

```

Протестуйте алгоритм декілька разів і порівняйте результати. Самостійно змініть діапазон вибору випадкових чисел в алгоритмі.

Завдання 3. Якщо протягом n років на депозитну картку з сумою K грн нараховується m разів щорічно деякий постійний складний відсоток P% , то заощадження обчислюється за формулою

$$S=K* exp(ln (1+p/(100*m))*n*m)$$

Скласти і реалізувати алгоритм для вивчення сум грошей на депозитній картці через декілька років.

```

program BABLO_3;           {назва алгоритму}

var k,s: real;    u,m,n, i: integer;    {оголошення змінних величин дійсні числа та цілі числа}

begin                   { початок виконання алгоритму і введення випадкових значень}

k:=1000+random(1000); writeln( ' Якщо початкова сума на депозиті k=', k); writeln;

p:=7+random(10); writeln( ' Якщо відсоткова ставка для цього депозиту p=', p);
writeln;

```

```

n:=1+random(5); writeln( ' Якщо кількість років існування депозиту n=', n); writeln;

m:=1+random(12); writeln( ' Якщо кількість нарахувань на депозит за рік m=', m);
writeln;

s:=k; u:=n*m;      {обчислення чисел для виконання циклу в алгоритмі}

fori:=1 to u do begin { цикл з лічильником по нарахуванням для обчислення грошей}

s:= K*exp(ln(1+p/(100*i))*i*i);

writeln( 'Кількість грошей на депозиті після', i, '-го нарахування S=', s); writeln;end;
writeln('*****'); end.

```

Протестуйте алгоритм декілька разів і порівняйте результати. Самостійно змініть діапазон вибору випадкових чисел в алгоритмі.

Практична робота 18.

Алгоритми повторення різних типів

Завдання 1. Створити та реалізувати алгоритм мовою програмування Pascal, що містить двовимірний масив **array [1 .. n, 1 .. 10]**, вкладені цикли **for i:=1 to n do** з лічильниками по двом індексам і цикл **while ... do** з передумовою для обчислення похибок обчислень в масиві.

```

program proced1;

const n=5; m=10;   type   mas = array [1 .. n, 1 .. m] of real;   var a: mas; e, eps:
real;   j,i: integer;

begin

for i:=1 to n do begin for j:=1 to m do a[i,j]:=random(10); end;   eps:=0.001;

for i:=1 to n do begin for j:=1 to m do begin e:= 0.987*a[i,j]-(1-0.123*eps)*a[i,j];

while e > 0.0001 do e:=e/10; end; end;

writeln('Це похибка обчислень', e); end.

```

Завдання 2. Створити та реалізувати алгоритм мовою програмування Pascal, що містить двовимірний масив **array [1 .. n, 1 .. 10]**, вкладені цикли **for i:=1 to n do** з лічильниками по двом індексам і цикл **repeat until ...** з післядумовою для обчислення похибок обчислень в масиві.

```

program proced2;

const n=3; m=4;   type   mas = array [1 .. n, 1 .. 10] of real;   var b: mas; e, eps:

```

```

real; j,i: integer;

begin

for i:=1 to n do begin for j:=1 to m do b[i,j]:=random(10); end;

eps:=0.001;

for i:=1 to n do begin for j:=1 to n do begin

repeat e:= a[i,j]-(1-eps)*a[i,j]; e:=e/10; until e>0.01; end; end;

writeln('Це похибка обчислень', e); end.

```

Завдання 3. Створити та реалізувати алгоритм мовою програмування Pascal, що містить **процедуру**, яка знаходить найбільше значення двох цілих чисел.

```

program proced3;

var x,y,m,n: integer;

    procedure maxnumber(a,b: integer; var max: integer);

    begin if a>b then max:=a else max:=b; end;

begin

x:=random(100); writeln(' x=', x); y:=random(100); writeln(' y= ', y);

maxnumber(x,y,m); maxnumber(2,x+y,n);

writeln(' max=', m, ' summa n=', n); end.

```

Завдання 4. Створити та реалізувати алгоритм, що містить **функцію**, яка знаходить найбільше значення двох цілих чисел.

```

program proced4;

var x,y,m,n: integer;

function MaxNumber(a,b: integer): integer;

    var max: integer;

begin

    if a>b then max:=a else max:=b;

    MaxNumber := max; end;

begin

```



```

x:=40-random(100);  writeln(' x=', x);  y:=50-random(100);    writeln(' y= ', y);

m := MaxNumber(x,y);    n := MaxNumber(2,x+y);

writeln('max=',m,' summa=',n); end.

```

Практична робота 19.

Алгоритми табуляції поліномів на мові Pascal

Завдання 1. Скласти і реалізувати алгоритм для знаходження табулювання значень квадратичної функції, нулів функції, що задана рекурсивною формулою $y=(a_1x+a_2)x+a_3= a_1x^2+a_2x+a_3$.

```

program Quadratform;           {назва алгоритму табуляції}

var a1, a2, a3, x, y, d: real;   i, k: integer;   {оголошення змінних величин: дійсні числа та цілі числа}

begin      {початок виконання алгоритму і введення випадкових коефіцієнтів квадратичної функції}

  a1:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 1+random(3) ;

  writeln(' Якщо старший коефіцієнт квадратичної функції a1=', a1); writeln;

  a2:=2+random(7) *(-1+random(3));

  writeln(' Якщо лінійний коефіцієнт квадратичної функції a2=', a2); writeln;

  a3:= (4+random(10))*(-1)*(-1+random(3));

  writeln(' Якщо вільний коефіцієнт квадратичної функції: y(0)=a3=', a3); writeln;

  writeln('Якщо сума коефіцієнтів квадратичної функції: y(1)=a1+a2+a3=', a1+a2+a3); writeln;

  x:=- (1+random(20)); writeln('Якщо початковий аргумент квадратичної функції x=', x); writeln;

  d:=1+random(3); writeln(' Якщо величина кроку табуляції квадратичної функції d =', d); writeln;

  k:=5+random(8); writeln(' Якщо кількість кроків табуляції k=', k); writeln;

  for i:=1 to k do begin      {виконання циклу з лічильником по крокам для обчислення значень функції}

```

$x:=x+(i-1)*d;$ $y:=(a1*x+a2)*x+a3;$ { виведення результатів табулювання на екран монітора }

writeln(' номер кроку табуляції i=', i, ' аргумент функції x=', x, ' значення функції y=', y); writeln; end; writeln('***');** **writeln(' квадратична функція має вигляд y=', a1, '*x*x+(', a2, ')x+(', a3, ')');**

d:=a2*a2-4*a1*a3;

if (d>0) or (d=0) then { обчислення дискримінанта перевірка його знаку для квадратичної функції }

begin $x:=0.5*(-a2-\text{sqrt}(d))/a1;$ $y:=\text{abs}((a1*x+a2)*x+a3);$ { обчислення першого нуля квадратичної функції }

writeln('перший наближений нуль квадратичної функції x1=', x, ' з абсолютною похибкою ',y); writeln;

$x:=0.5*(-a2+\text{sqrt}(d))/a1;$ $y:=\text{abs}((a1*x+a2)*x+a3);$ { обчислення другого нуля квадратичної функції }

writeln('другий наближений нуль квадратичної функції x2=', x, ' з абсолютною похибкою ',y); writeln; end

else **writeln(' немає нулів квадратична функція');**

$x:=0.5*(-a2)/a1;$ $y:=(a1*x+a2)*x+a3;$ { обчислення координат вершини параболи }

if (a1>0) then { оголошення мінімуму квадратичної функції як координат вершини параболи }

writeln(' наближений мінімум квадратичної функції якщо Xmin=', x, ' та Ymin= ', y)

else { оголошення максимуму квадратичної функції як координат вершини параболи }

writeln(' наближений максимум квадратичної функції якщо Xmax=', x, ' та Ymax= ',y); writeln;

writeln(' немає перегинів випуклих ділянок квадратична функція');

end. { закінчення алгоритму }

Протестуйте алгоритм чотири рази та порівняйте результати табуляції і виберіть той варіант, при якому можна знайти найточніше наближення нулів квадратичної функції, тобто випадок $y:=(a1*x+a2)*x+a3=0$.

Завдання 2. Скласти і реалізувати алгоритм для знаходження табулювання значень кубічної функції, що задана рекурсивною формулою $y=((a_1x+a_2)x+a_3)x+a_4== a_1x^3+ a_2x^2+a_3x+a_4$.

```

program Cubfunction;           {назва алгоритму табуляції}

var a1, a2, a3, a4, x, y, d: real;   i, k: integer;   {оголошення змінних величин: дійсні
числа та цілі числа}

begin           {початок виконання алгоритму і введення випадкових коефіцієнтів кубічної
функції}

  a1:=1+random(3); writeln( ' Якщо старший коефіцієнт кубічної функції a1=', a1);
  writeln;

  a2:=2-random(7); writeln(' Якщо квадратичний коефіцієнт кубічної функції a2=', a
  2); writeln;

  a3:=- (4-random(10)); writeln(' Якщо лінійний коефіцієнт кубічної функції a3=', a3);
  writeln;

  a4:=- (3-random(10)); writeln(' Якщо вільний коефіцієнт кубічної функції a4=', a4);
  writeln;

  x:=- (1+random(20)); writeln(' Якщо початковий аргумент кубічної функції x=', x);
  writeln;

  d:=1+random(3); writeln( ' Якщо величина кроку табуляції кубічної функції d=', d);
  writeln;

  k:=15+random(5); writeln( ' Якщо кількість кроків табуляції k=', k); writeln;

  for i:=1 to k do begin           {виконання циклу з лічильником по крокам для обчислення
значень функції}

    x:=x+(i-1)*d;   y:=((a1*x+a2)*x+a3)+a4;   { виведення результатів табулювання на
екран монітора}

    writeln( ' номер кроку табуляції i=', i, ' аргумент функції x=', x, ' значення функції
y=', y); writeln; end; writeln('*****'); end.   {закінчення алгоритму}

```

Протестуйте алгоритм чотири рази та порівняйте результати табуляції і виберіть той варіант, при якому можна знайти найточніше наближення нулів кубічної функції, тобто випадок $((a1*x+a2)*x+a3)*x+a4=0$.

Практична робота 20.

Табулювання кускових функцій

Завдання 1. Створити та реалізувати мовою Паскаль алгоритм, який обчислює значення функції при заданому аргументі за такими формулами:



Алгоритм використовує послідні розгалуження в повній формі.

programtabuljacia1;

var f,x,d: real; n, i: integer; {оголошуються змінні типу: дійсні та цілі}

begin writeln('вводиться початкове випадкове число, яке є аргументом функції ');

x:=- (10+random(10)); write(x); writeln; n:= 10+random(20); d:= 1+random(5);

write('якщо кроків n =', n); writeln; write('якщо довжина кроку d =', d); writeln;

for i:=1 to n do begin {виконується цикл з лічильником для обчислення функції }

if x<-10 then f:=x*x else {розгалуження для перевірки аргументу функції }

if x<10 then f:=x else f:=-x*x; x:=x+d; { розгалуження та обчислення }

writeln('результат обчислення f(' x, ')=' f, 'якщо номер кроку i =', i); writeln; end;
end.

Протестуйте цей алгоритм декілька разів і порівняйте результати обчислень на кожному кроці. Спробуйте самостійно змінити в алгоритмі діапазон випадкових чисел.

Завдання 2. Створити та реалізувати мовою Паскаль алгоритм, який обчислює значення кускової функції.

programtabuljacia2;

var p, n, i: integer; {оголошуються змінні типу: цілі}

beginwriteln('вводиться початкове випадкове число, яке є аргументом функції ');

p:=- (10+random(10)); write(p); writeln; n:= 10+random(20);

write('якщо кроків n =', n); writeln;

for i:=1 to n do begin {виконується цикл з лічильником для обчислення функції }

if (p = 0) or (p = 1) then p:=0 else {розгалуження для перевірки аргументу функції }

if (p mod 2 = 0) then p:=p div 2 else p:=3*p+1; { розгалуження та обчислення }

writeln('результат обчислення

p(' i, ')=' p, 'якщо номер кроку i =', i); writeln; end; end.

Протестуйте цей алгоритм декілька разів і порівняйте результати обчислень на кожному

кроці. Спробуйте самостійно змінити в алгоритмі діапазон випадкових чисел.

Завдання 3. Створити та реалізувати мовою Паскаль алгоритм, який обчислює значення функції.

program tabuljacia3;

var g, n, i: integer; {оголошуються змінні типу: цілі}

beginwriteln('вводиться початкове випадкове число, яке є аргументом функції ');

g:=-(10+random(10)); **write(g); writeln; n:= 10+random(20);**

write('якщо кроків n =', n); writeln;

for i:=1 to n do begin {виконується цикл з лічильником для обчислення функції }

if (g mod 3 = 0) then g:=3*g+1 else {розгалуження перевірки аргументу функції }

if (g mod 3 = 1) then g:=g-2 div 2 else g:=3*g+2; {розгалуження та обчислення}

writeln('результат обчислення g(', i, ')=' , g, 'якщо номер кроку i =', i); writeln; end;
end.

Протестуйте цей алгоритм декілька разів і порівняйте результати обчислень на кожному кроці. Спробуйте самостійно змінити в алгоритмі діапазон випадкових чисел.

Практична робота 21.

Алгоритми інтерполяції функції

Завдання 1. Створити алгоритм який за трьома відомими точками в прямокутній системі координат генерує формулу квадратичної функції використовуючи розв'язання системи 3-х рівнянь з трьома невідомими методом Крамера. Ця задача називається «інтерполяція квадратичними поліномами» або знаходження «квадратичного тренду».

Розв'язання. Випадковим чином задаються три точки деякої статистики: $(x_1; y_1)$, $(x_2; y_2)$, $(x_3; y_3)$. Вважається, що ці три точки належать деякій параболі, що записується формулою вигляду: $y=ax^2+bx+c$. Підставляємо кожну точку у формулу і отримуємо систему трьох рівнянь з трьома невідомими a, b, c .

$$\{ | \begin{matrix} ax_1^2 + bx_1 + c = y_1 \\ ax_2^2 + bx_2 + c = y_2 \\ ax_3^2 + bx_3 + c = y_3 \end{matrix} | \}$$

$$\{ | ax^2*x^2 + bx^2 + c * 1 = y_2 \}$$

$$\{ | ax^3*x^3 + bx^3 + c * 1 = y_1 \}$$

Розв'язуємо систему відносно a, b, c за допомогою метода визначників (метод Крамера).

Program Interpoljacia;

```
var a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, x, y, z, e, ex, ey, ez, x1, x2, x3, y1, y2, y3:
real;
```

```
begin
```

```
x1:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 1+random(3) ;
```

```
y1:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 2+random(3) ;
```

```
writeln ( ' Якщо перша точка, що належить квадратичній функції x1=', x1, 'y1=', y1);
writeln;
```

```
x2:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 1+random(3) ;
```

```
y2:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 3+random(3) ;
```

```
writeln ( 'Якщо друга точка, що належить квадратичній функції x2=', x2, 'y2=', y2);
writeln;
```

```
x3:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 1+random(3) ;
```

```
y3:=- (1+random(3))*(-2+random(1)) *(-random(2)) + 4+random(3) ;
```

```
writeln ( ' Якщо третя точка, що належить квадратичній функції x3=', x3, 'y3=', y3);
writeln;
```

```
a1:= x1*x1; a2:= x2*x2; a3:= x3*x3; b1:= x1; b2:= x2; b3:= x3; c1:=1; c2:=1;
c3:=1;
```

```
d1:=y1; d2:=y2; d3:=y3;
```

```
e:= (a1 * b2 * c3 + b1 * c2 * a3 + c1 * a2 * b3-a3 * b2 * c1-b3 * c2 * a1-c3 * a2 * b1);
```

```
ex:=(d1 * b2 * c3 + b1 * c2 * d3 + c1 * d2 * b3-d3 * b2 * c1-b3 * c2 * d1-c3 * d2 * b1);
```

```
ey:=(a1 * d2 * c3 + d1 * c2 * a3 + c1 * a2 * d3-a3 * d2 * c1-d3 * c2 * a1-c3 * a2 * d1);
```

```
ez:=(a1 * b2 * d3 + b1 * d2 * a3 + d1 * a2 * b3-a3 * b2 * d1-b3 * d2 * a1-d3 * a2 * b1);
```

```
if (e=0) and ((ex=0) or (ey=0) or (ez=0)) then
```

```
writeln ( 'безліч рішень')
```

```
else if (e <> 0) and ((ex = 0) or (ey = 0) or (ez = 0)) then
```

```
writeln ( 'немає рішень')
```

else begin

x:=ex/e; y:=ey/e; z:=ez/e;

writeln ('Головний визначник e =', e);writeln ('a =', x); writeln ('b =', y); writeln ('c =', z);

writeln('Шукана квадратична функція y=', x, '*x*x+(', y, ')x+(', z, ') '); end; end.

Протестуйте правильність виконання алгоритму.

Практична робота 22.

Алгоритми розгалуження та вибору.

Завдання 1. Створити алгоритм повного розгалуження **if then ... else**

program przyklad_01;

var a,b,c,y: real;

begin

write('Vvedit a'); readln(a);

write('Vvedit b'); readln(b);

write('Vvedit c'); readln(c);

if a>=15 then

begin

if a<=20 then begin writeln('a v diapazoni 15-20 ');

y:=10-(exp(abs(a-b))*(exp(a*(ln(sqr(sin(c)/cos(c)+1))))); end

else writeln begin ('a >20 ');

y:=20-(exp(abs(a-b))*(exp(a*(ln(sqr(sin(c)/cos(c)+1))))); end;

end

else writeln('a<15');

y:=30-(exp(abs(a-b))*(exp(a*(ln(sqr(sin(c)/cos(c)+1)))));

```
writeln('y=',y);
```

```
readln;
```

```
end.
```

Завдання 2. Створити алгоритм повного вибору **case N of else**

```
Program zrazok_02;
```

```
var Num: integer;
```

```
begin
```

```
write('Введіть число:');
```

```
readln(Num);
```

```
case Num of
```

```
0: writeln('Нуль');
```

```
1: writeln('Один');
```

```
2: writeln('Два' );
```

```
3: writeln('Три ');
```

```
4: writeln('Чотири') ;
```

```
5:writeln('П'ять');
```

```
6:writeln('Шість');
```

```
7:writeln('Сім');
```

```
8:writeln('Вісім');
```

```
9:writeln('Дев'ять');
```

```
else writeln('Число не є цифрою'); end; readln; end.
```

Завдання 3. Створити алгоритм повного розгалуження **if then ... else**

```
program Bilshe03;
```

```
var a,b:integer;
```

```
begin
```

```
  writeln('Vvedit a:'); readln(a);
```

```
  writeln('Vvedit b:'); readln(b);
```

```
  if a>b then writeln('a=',a) else
```

```
    if a<b then writeln('b=',b) else writeln('a=b'); end.
```

Практична робота 23.

Алгоритми розгалуження геометричного змісту

Завдання 1. Створити та реалізувати мовою програмування Pascal, що визначає в скількох точках перетинаються два кола за шістьма чисел $x_1, y_1, r_1, x_2, y_2, r_2$, де x_1, y_1, x_2, y_2 - координати центрів кіл, r_1, r_2 - їх радіуси. Усі числа - дійсні, не перевищують 1000000000 за

модулем, та задані не більш ніж із 3 знаками після коми.

Program Circus01;

var x1,y1,r1,x2,y2,r2, d:real;

begin

readln (x1,y1,r1,x2,y2,r2); d:=sqr(x1-x2)+sqr(y1-y2);

if (x1=x2) and (y1=y2) and (r1=r2) then writeln (-1)

else if (sqr(r1+r2)=d) or (sqr(r1-r2)=d) then writeln (1)

else if (sqr(r1+r2)<d) or (sqr(r1-r2)>d) then writeln (0)

else writeln (2); end.

Завдання 2. Створити та реалізувати мовою програмування Pascal, що визначає

за розмірами прямокутних дверей **a, b** та розмірами шафи, що має форму прямокутного паралелепіпеда **x, y, z, x, y, z < 10** чи можна пронести шафу у двері, якщо пронести її дозволяється так, щоб кожне ребро шафи було паралельне або перпендикулярне кожній стороні дверей.

program SHAFa2;

var a, b, c, x, y, z:real;

begin readln (a, b, x, y, z);

if ((x<a) and (y<b)) or ((x<b) and (y<a)) or ((z<a) and (y<b)) or

((x<a) and (z<b)) or ((y<a) and (z<b)) or ((z<a) and (x<b))

then writeln (1) else writeln (0); end.

Завдання 3. Створити та реалізувати алгоритм мовою програмування Pascal.

Гена збирається на туристичний зліт учнів своєї школи. У своєму класі його було призначено відповідальним за палатки. У себе вдома він знайшов 3 палатки: перша з них важить **a1** кілограм і вміщує **b1** чоловік, друга важить **a2** кілограм і вміщує **b2** чоловік, третя важить **a3** кілограм і вміщує **b3** чоловік.

У класі Гени **k** чоловік. Виясніть, чи може він вибрати палатки так, щоб у них змогли поміститись усі. При цьому враховуйте, що вибрані палатки повинні разом важити не більше **w** кілограм.

Вхідні дані: два цілих числа **k** та **w** ($1 \leq k \leq 15$, $1 \leq w \leq 30$). Другий рядок містить шість цілих чисел: **a1, b1, a2, b2, a3, b3** ($1 \leq a1, a2, a3 \leq 10$, $1 \leq b1, b2, b3 \leq 15$).

program KLaSS4;

```

var k,w,a1,a2,a3,b1,b2,b3: integer;

n1, n2, n3, n4, m1, m2, m3, m4: integer;

begin read (k,w); readln (a1,b1,a2,b2,a3,b3);

m1 := a1 + a2; n1 := b1 + b2; m2 := a1 + a3; n2 := b1 + b3;

m3 := a3 + a2; n3 := b3 + b2; m4 := a1 + a2 + a3; n4 := b1 + b2 + b3;

if ( (a1<=w) and (b1>=k) ) or ( (a2<=w) and (b2>=k) ) or
( (a3<=w) and (b3>=k) ) or ( (m1<=w) and (n1>=k) ) or
( (m2<=w) and (n2>=k) ) or ( (m3<=w) and (n3>=k) ) or
( (m4<=w) and (n4>=k) ) then writeln('YES') else writeln('NO');

end.

```

Завдання 4. Створити та реалізувати алгоритм мовою програмування Pascal.

У різдвяний вечір на підвіконні стояли три квіточки, зліва направо: герань, крокус та фіалка. Кожен ранок Маша витирала підвіконня і міняла місцями квіточку, що стояла праворуч, з центральною квіточкою. А Таня кожен вечір поливала квіточки і міняла місцями ліву та центральну квіточки. Потрібно визначити порядок квітів вночі після того, як пройде **k** днів.

Вивести **m** рядків, що містять по три латинських літери: "G", "C" и "V" (великі літери без пропусків), які описують порядок квітів на вікні по закінченню **k** днів (зліва направо). Позначення: G - герань, C - крокус, V - фіалка.

```

program E4;

var m, k, i: integer;

begin readln(m);

for i := 1 to m do begin readln(k);

if (k mod 3 = 1) then writeln('VGC');

if (k mod 3 = 2) then writeln('CVG');

if (k mod 3 = 0) then writeln('GCV');

end; end.

```

Практична робота 24.

Алгоритми розгалуження і повторення

Завдання 1. Створити та реалізувати алгоритм мовою програмування Pascal.

Степан влітку відпочиває у бабусі в селі. Особливо йому подобається

купатись на сільському озері. Посередині озера плаває пліт, який має форму прямокутника. Сторони плота спрямовані уздовж паралелей і меридіанів. Введемо систему координат, в якій вісь OX направлена на схід, а вісь OY – на північ. Нехай південно-західний кут плота має координати (x_1, y_1) , північно-східний кут - координати (x_2, y_2) .

Степан знаходиться в точці з координатами (x, y) . Визначте, до якої сторони плота (північної, південної, західної чи східної) або до будь-якого кута плота (північно-західному, північно-східному, південно-західному, південно-східному) Степану потрібно плисти, щоб якомога швидше дістатися до плота.

Формат вхідних даних:

Дано шість чисел в наступному порядку: x_1, y_1 (координати південно-західного кута плота), x_2, y_2 (координати північно-східного кута плота), $x,$

y (координати Степана). Всі числа цілі і по модулю не перевершують 100.

Гарантується, що $x_1 < x_2, y_1 < y_2, x \neq x_1, x \neq x_2, y \neq y_1, y \neq y_2$, координати Степана знаходяться поза плотом.

Формат вихідних даних:

Якщо Степану слід плисти до північної сторони плота, програма повинна вивести символ «N», до південної - символ «S», до західної - символ «W», до східної - символ «E». Якщо Степану слід плисти до кута плота, потрібно вивести один з наступних рядків: «NW», «NE», «SW», «SE».

Пліт умовно розбиває площину на 8 частин. (див. мал.)



Очевидно, якщо точка з координатами (x, y) знаходиться у 2, 4, 6 або 8

частині площини, то найкоротшою відстанню до плота буде перпендикуляр, проведений відповідно до північної, східної, південної або західної сторони.

Якщо точка з координатами (x, y) знаходиться у 1, 3, 5, 7 частинах

площини, то найкоротшою відстанню буде відповідний кут плота.

Розв'язання:

Program PROBA_01;

var x1,x2,x,y1,y2,y:integer;

begin

read(x1,y1,x2,y2,x,y);

if (x<x1) and (y>y1) and (y<y2) then write('W');

if (y>y2) and (x>x1) and (x<x2) then write('N');

if (x>x2) and (y>y1) and (y<y2) then write('E');

if (y<y1) and (x>x1) and (x<x2) then write('S');

if (x<x1) and (y>y2) then write('NW');

if (x>x2) and (y>y2) then write('NE');

if (x<x1) and (y<y1) then write('SW');

if (x>x2) and (y<y1) then write('SE');

end.

Завдання 2. Створити та реалізувати мовою програмування Pascal, що визначає

найбільшу кількість олівців k, які може купити Семен на S гривень після подорожчання на P відсотків.

Технічні умови. У першому рядку задано число N ($1 \leq N \leq 10^7$) - вартість олівця до подорожчання. У другому рядку - P ($0 \leq P \leq 100$) - величина подорожчання олівця у відсотках. В третьому рядку - S ($1 \leq S \leq 10^7$) - сума грошей, яка є у Семена.

Program PROBA_02;

var n,s,p,k,t:integer;

begin

read(n); readln(p); readln(s);

k:= s*100 div(n+n*p); ;

write(k);

end.

Завдання 3. Створити та реалізувати мовою програмування Pascal, що визначає вагу запакованих речей рюкзака і валізи, якщо Семен збирає речі у похід. З собою в автобус він може взяти ручну поклажу і багаж. Для ручної поклажі у нього є рюкзак, а для багажу - здорова валіза. За правилами перевезення маса ручної поклажі не повинна перевищувати S кг, а багаж може бути будь-якої маси (за наднормативний багаж Семен готовий доплатити). Зрозуміло, найбільш цінні речі - ноутбук, фотоапарат, документи і т. д. - Семен хоче покласти в ручну поклажу.

Семен розклав усі свої речі в порядку зменшення їх цінності і починає складати найбільш цінні речі в рюкзак. Він діє в такий спосіб - бере найцінніший предмет, і якщо його маса не перевищує S , то кладе його в рюкзак, інакше кладе його до валізи. Потім він бере наступний за цінністю предмет, якщо його можна покласти в рюкзак, тобто якщо його маса разом з масою вже покладених в рюкзак речей не перевищує S , то кладе його в рюкзак, інакше до валізи, і таким же чином процес триває для всіх предметів в порядку спадання їх цінності.

Визначте вагу рюкзака і валізи після того, як Семен складе всі речі.

Формат вхідних даних:

Перший рядок вхідних даних містить число S ($1 \leq S \leq 2 \times 10^9$) - максимально дозволена вага рюкзака. У другому рядку вхідних даних записано число N ($1 \leq N \leq 10^5$) - кількість предметів.

У наступних N рядках дано маси предметів, самі предмети перераховані в порядку спадання цінності (спочатку вказана маса найціннішого предмета, потім маса другого по цінності предмета і т. Д.). Всі числа натуральні, сума ваги всіх предметів не перевищує 2×10^9 .

Формат вихідних даних:

Виведіть два числа - вагу рюкзака і вагу валізи (вага порожнього рюкзака і валізи не враховується).

Приклади Вхідні дані	Результат роботи
10 4 6 3 4 1	10 4

Нехай $m1$ - вага рюкзака і $m2$ - вага валізи.

Беремо перший предмет з вагою a і перевіряємо, якщо $m1+a$ не перевищує s , то збільшуємо $m1$ на a , інакше $m2$ збільшуємо на a .

Розв'язання: 1 спосіб

Program PROBA_03;

```

var s,n,a, m1, m2,i:integer;

begin

readln(s); readln(n);

i:=1; m1:=0; m2:=0;

while i<=n do begin readln(a);

if (m1+a<=s) then m1:=m1+a else m2:=m2+a;

i:=i+1; end;

write(m1,' ',m2); end.

```

2 спосіб

```

var v,b,n,s,i:longint;  a:array [1..1000000] of longint;

begin

read(s,n);

for i:=1 to n do read(a[i]);

b:=0; v:=0;

for i:=1 to n do

  if b+a[i]<=s then b:=b+a[i] else v:=v+a[i];

write(b,' ',v);

end.

```

Завдання 4. Степан виписує на листочку усі цілі числа від 1 до N в кілька груп, при цьому якщо одне число ділиться на інше, то вони обов'язково будуть у різних групах.

Наприклад, якщо $N = 9$, то отримаємо 4 групи:

Перша група: 1.

Друга група: 2 3 7.

Третя група: 4 5 6.

Четверта група: 8 9.

Очевидно, що оскільки, будь-яке число ділиться на 1, то одна група завжди буде складатись тільки з числа 1, а от інші групи можуть бути створені різними способами.

Допоможіть Степану написати програму, яка визначає мінімальне число груп, на яке можна розбити усі числа від 1 до N у відповідності до наведеної вище умови.

Формат вхідних даних:

Перший рядок вхідних даних містить єдине число N ($1 \leq N \leq 109$).

Формат вихідних даних:

Виведіть одне число - знайдену мінімальну кількість груп.

Приклади Вхідні дані	Результат роботи
9	4

Наприклад, якщо визначати кількість груп для послідовних чисел від 1 до 20, то можна побачити, що вона (кількість) є степенем числа 2. Таким чином для будь-якого числа N можна розглянути нерівність $2^k \leq N < 2^{k+1}$, і шуканою відповіддю буде число $k+1$.

Program proba04;

```
var n,v,i,st: integer;
```

```
begin
```

```
read(n);
```

```
v:=1; st:=2; i:=2;
```

```
while i<=n do begin i:=st*2; st:=i; inc(v); end;
```

```
if n<>1 then write(v) else write(1); end.
```