Міністерство освіти і науки України.

Львівський національний університет імені Івана Франка.

Звіт з навчальної практики.

Виконав студент

групи ПМп-11

Качор Михайло

Львів-2015

Завдання №1

1. Умова завдання.

Для заданого  побудувати табличку вигляду

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | ... | n-1 | n |
| n | 1 | 2 | ... | n-2 | n-1 |
| n-1 | n | 1 | ... | n-3 | n-2 |
| ... | ... | ... | ... | ... | ... |
| 2 | 3 | 4 | ... | n | 1 |

1. Алгоритм розв’язання.

Крок 1. Перевіряємо в котрій частинні матриці знаходиться елемент.

Крок 2. Заповнюємо праву частино матриці відносно головної діагоналі за формулою a[i,j]=j-i+1

Крок 3. Заповнюємо ліву частино матриці відносно головної діагоналі за формулою a[i,j]=j-i+1+n;

1. Програмна реалізація.

**program** No1;

**const**

n = 10;

**type**

mas = **array**[1..n, 1..n]**of** integer;

**var**

i, j: integer;

q: mas;

**begin**

**for** i := 1 **to** n **do**

**for** j := 1 **to** n **do**

**begin**

**if** (j >= i) **then**

q[i, j] := j - i + 1

**end**;

**for** i := 1 **to** n **do**

**for** j := 1 **to** n **do**

**if** (j < i) **then**

q[i, j] := j - i + n + 1;

**for** i := 1 **to** n **do**

**begin**

writeln;

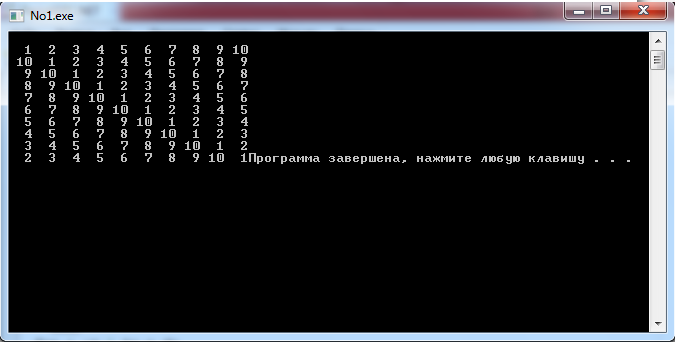
**for** j := 1 **to** n **do**

write(q[i, j]:3);

**end**;

**end**.

1. Апробація програми.

****

Завдання №2

1. Умова завдання.

Дано масив з n натуральних чисел. Перевірити чи даний масив є перестановкою.(тобто містить всі числа від 1 до n в довільному порядку)

1. Алгоритм розв’язання.

Крок 1. Перевіряємо почергово від 1 до n чи в масиві міститься дане число.

Крок 2. Якщо так продовжуємо до першого не знайденого елемента.

Крок 3. Якщо всі елементи знайденні масив є перестановкою.

1. Програмна реалізація.

**Program** Task2;

**type**

mas=**array**[1..100] **of** integer;

**var**

i, n, j, counter:integer;

a:mas;

**Begin**

write('Please enter number of elements of array: ');

readln(n);

write('Please enter elements of array: ');

**for** i:=1 **to** n **do**

**begin**

read(a[i]);

**end**;

**for** i:=1 **to** n **do**

**begin**

counter:=0;

**for** j:=1 **to** n **do**

**begin**

**if** a[j]=i **then**

**begin**

inc(counter);

**end**;

**end**;

**if** counter=0 **then**

**begin**

writeln('Array is NOT a transposition');

**exit**;

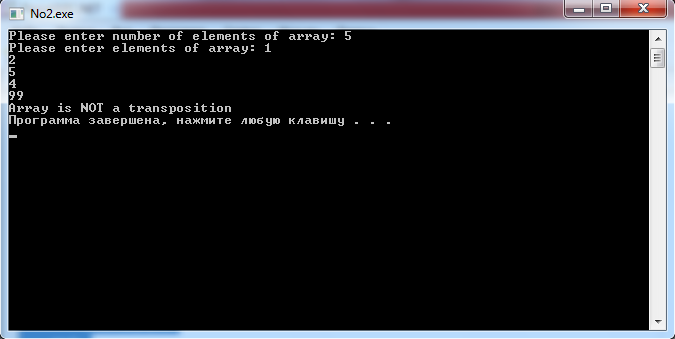
**end**;

**end**;

writeln('Array IS a transposition');

**end**.

1. Апробація програми.



Завдання №3

1. Умова завдання.

Прочитати дійсну матрицю розміром n×n (n > 5). Знайти скалярний добуток рядка, у якого j–ий елемент дорівнює ***a***, та стовпця, у якого i–ий елемент дорівнює ***b***. (якщо такого рядка (стовпця) нема, то взяти i-ий рядок (j-ий стовпець)).

1. Алгоритм розв’язання.

Крок 1.Шукаєм рядки в яких є значення параметра а і записуємо в матрицю.

Крок 2.Шукаєм стовпці в яких є значення параметра б і записуємо в матрицю.

Крок 3.Якщо два елементи матриць не 0 тоді шукаємо скалярний добуток.

1. Програмна реалізація.

**const** n=6;

**type** mass=**array**[1..n,1..n]**of** integer;mas=**array**[1..n]**of** integer;

**var** x:mass; v,a,b,i,j,kk:integer;h,y:boolean; o,z:mas;

**begin**

write('Vedit A=');

read(a);

write('Vedit B=');

read(b);

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

x[i,j]:= random(10);

**for** i:=1 **to** n **do**

**begin**

writeln;

**for** j:=1 **to** n **do**

write(x[i,j]:3);

**end**;

h:=false;

y:=false;

**begin**

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

**if** x[i,j]=a **then**

**begin**

o[i]:=i;

h:=true;

**end**;

**if** h=false **then**

o[n]:=n;

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

**if** x[i,j]=b **then**

**begin**

z[j]:=j;

y:=true;

**end**;

**if** y=false **then**

z[n]:=n;

writeln;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** n **do**

**begin**

**if** (z[i]<>0) **and** (o[j]<>0) **then**

**begin**

kk:=0;

**for** v:=1 **to** n **do**

kk:=kk+x[o[j],v]\*x[v,z[i]];

writeln;

write('Скалярний добуток:',kk);

**end**;

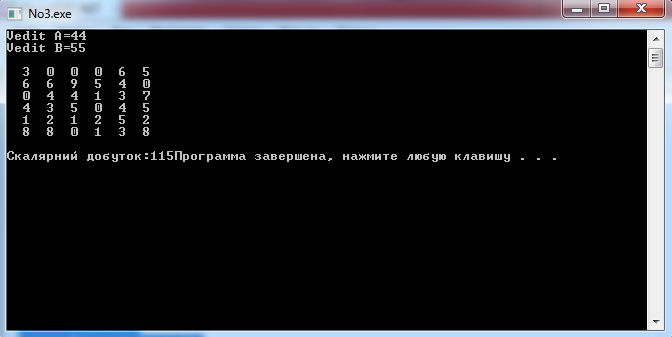
**end**;

**end**;

**end**;

**end**.

1. Апробація програми.



Завдання №4

1. Умова завдання.

Дано рядок, що складається зі слів, які розділені комами (однією або кількома). Створити і вивести рядок, що містить ці ж слова, розділені одним символом «.» (Крапка). В кінці рядка крапку не ставити.

1. Алгоритм розв’язання.

Крок 1.Замінюємо всі коми крапками.

Крок 2.Видаляємо всі крапки які повторюються.

Крок 3.Перевіряємо чи останній символ крапка якщо так видаляємо.

1. Програмна реалізація.

**program** p2;

**var**

t: string;

i, j: integer;

**begin**

writeln('Vvedit stroky');

read(t);

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

**begin**

**if** t[i] = ',' **then**

t[i] := '.';

**end**;

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

**for** j := 1 **to** length(t) - 1 **do**

**if** (t[j] = '.') **and** (t[j + 1] = '.') **then**

**begin**

delete(t, j, 1);

**break**;

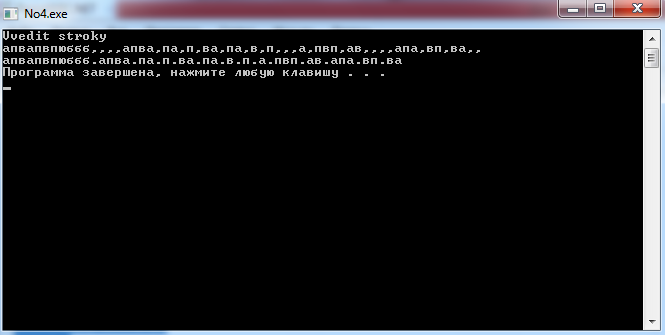
**end**;

**if** t[length(t)] = '.' **then** delete(t, length(t), 1);

writeln(t);

**end**.

4.Апробація програми.



Завдання №5

1. Умова завдання.

День вчителя відзначають в першу неділю жовтня. Дано натуральне число n≥2000, яке означає номер року. Визначити число, на яке число припаде день вчителя для даного року.

1. Алгоритм розв’язання.

Крок 1.Шукаємо різницю років між заданим та 2000.

Крок 2.Обчислюємо кількість високосних років.

1. Програмна реалізація.

**program** p2;

**var**

i, j, k, l, f, g: integer;

**begin**

writeln('Vvedit rik:');

read(k);

l := k - 2000;

f := l;

g := l **div** 4;

f := f + g;

**while** f >= 7 **do**

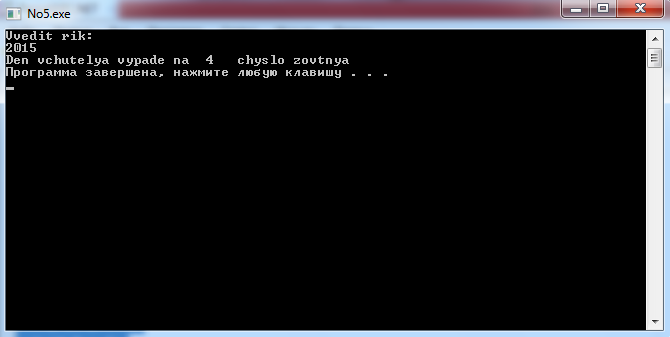
f := f - 7;

j := 8 - f;

writeln('Den vchutelya vypade na ', j, ' chyslo zovtnya');

**end**.

1. Апробація програми.



Завдання №6

1. Умова завдання.

Обчислити значення виразу 

1. Алгоритм розв’язання.

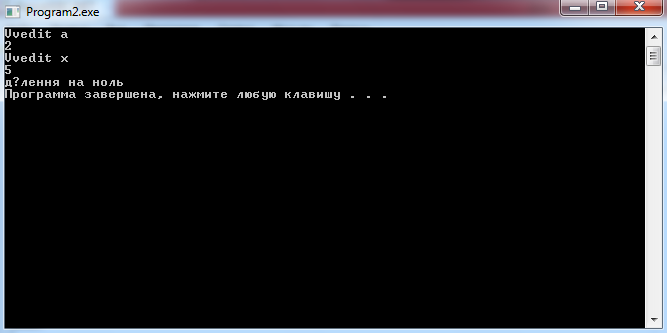
Крок 1.Обчислюємо значення виразу.

Крок 2.Обробляємо виключення за допомогою операторів exept try.

1. Програмна реалізація.

Program noname;  
var x,a,c:real;  
Function tag(x,a:real):real;  
begin  
tag:=sqrt(sqr(x)-x+3/(a-2)\*ln(a/x)/lna);  
end;  
begin  
try  
read(x,a);  
c:=tag(x,a);  
write(c);  
except  
on EZeroDivide do  
writeln('ділення на 0');  
on EInvalidArgument do  
writeln('від"єме число під коренем');  
end;  
end.

1. Апробація програми.



Завдання №7-8

1. Умова завдання.

Дано ціле число K і текстовий файл. Видалити з файлу рядки початкового файлу, номери яких кратні K. Якщо таких рядків нема, то залишити файл без змін.

Дано текстовий файл, кожен рядок якого зображує ціле число, доповнене ліворуч і праворуч декількома пропусками. Обчислити кількість цих чисел і їх суму.

1. Алгоритм розв’язання.

1)

Крок 1.Створюємо новий файл(програмно).

Крок 2.Копіюємо рядки кратності n.

Крок 3.Записуєм в основний файл інформацію з основного.

Крок 4.Видаляємо додатковий файл(програмно).

2)

Крок 1.Зчитуємо значення з файлу.

Крок 2.Обчислюємо суму і кількість.

1. Програмна реалізація.

1) **program** p3;

**var** t,f,a,b :text;

k:integer;

g:string;

i:integer;

**begin**

assign(t,'sim1.txt');

reset(t);

assign(f,'visim\_.txt');

rewrite(f);

writeln('Vvedit chyslo kratnocti');

read(k);

i:=0;

**while not** eof(t) **do**

**begin**

i:=i+1;

readln(t,g);

**if** (i **mod** k) =0 **then**

writeln(f,g);

**end**;

close(t);

close(f);

assign(a,'sim\_.txt');

rewrite(a);

assign(b,'visim\_.txt');

reset(b);

**while not** eof(b) **do**

**begin**

readln(b,g);

writeln(a,g);

**end**;

close(a);

close(b);

erase(b);

**end**.

**2)**

**program** gdsg;

**var**

t: text;

sum, kil, x: integer;

**begin**

assign(t, 'sim2.txt');

reset(t);

kil := 0;

sum := 0;

**while not** eof(t) **do**

**begin**

read(t, x);

sum := sum + x;

inc(kil);

**end**;

write('Suma elementiv= ');

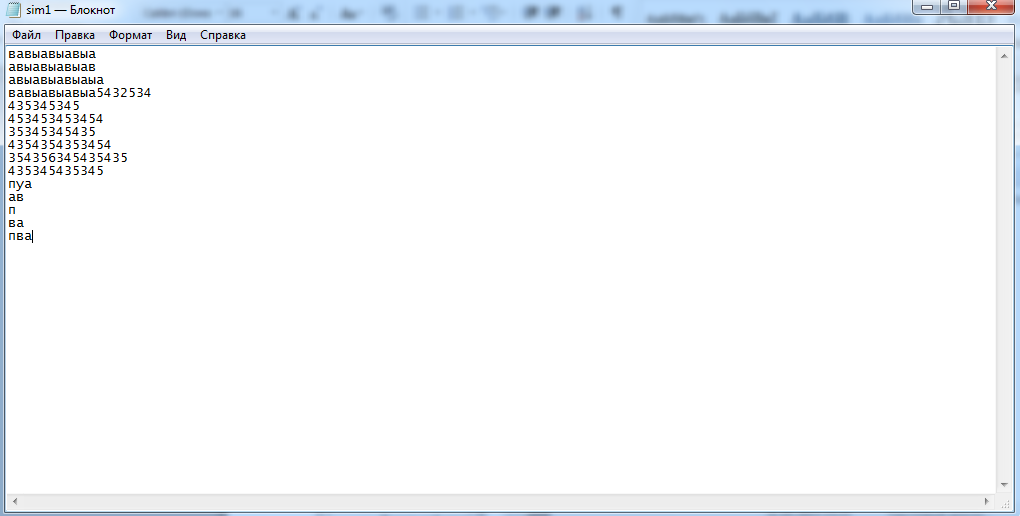
writeln(sum);

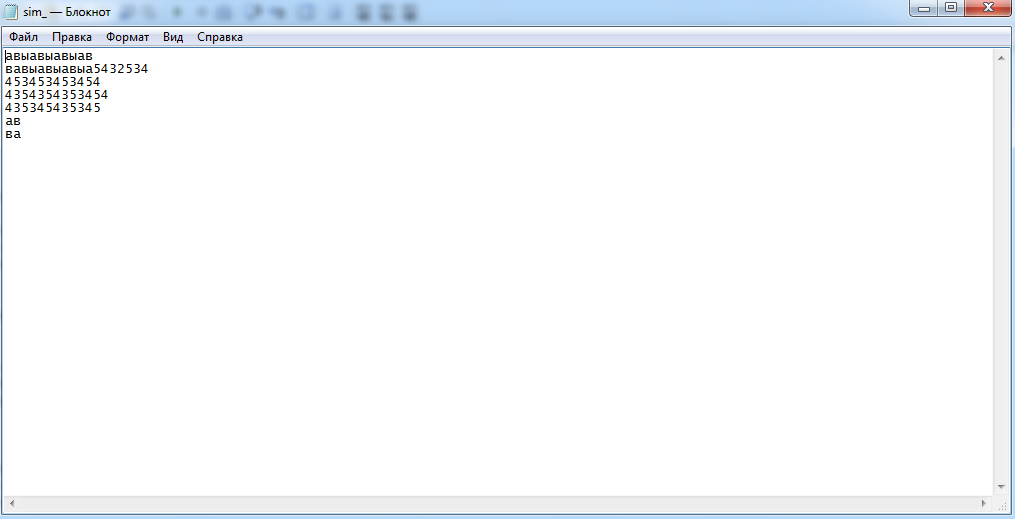
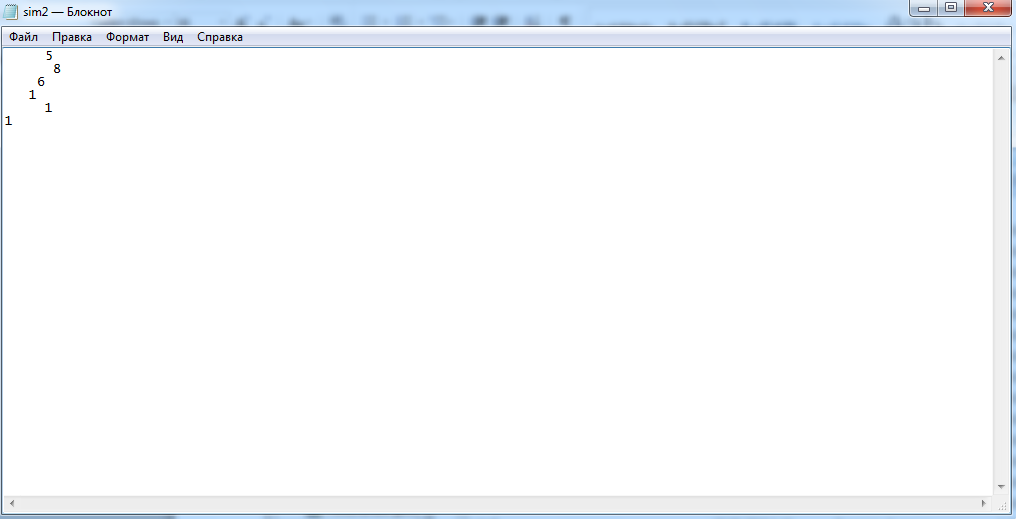
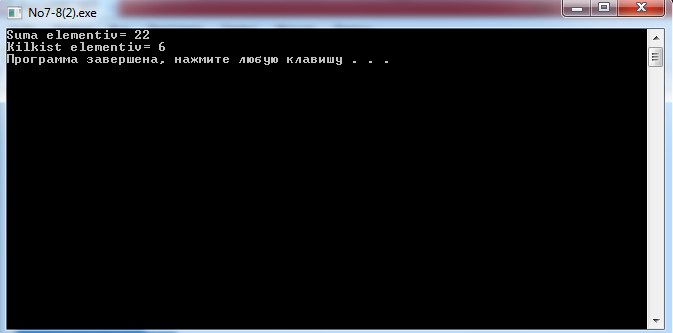
write('Kilkist elementiv= ');

writeln(kil);

**end**.

1. Апробація програми.

1)

2)

Завдання №9

1. Умова завдання.

Зобразити поліном  у вигляді списку. Якщо , то таку ланку в список не вносити. Описати функції або процедури для виконання наступних завдань:

1. Надрукувати поліном *р* від змінної *х* та його похідну (*p(х)=5x^5-2.5x^4+9*)
2. Обчислити значення полінома і його похідної.
3. Алгоритм розв’язання.

Крок 1.Створюємо список розбиваючи поліном на коефіцієнт та степінь.

Крок 2. Створюємо список з значеннями похідної.

Крок 3.Обчислюємо значення полінома та похідної.

1. Програмна реалізація.

**program** Project\_Practychna\_9;

**type**

list = ^lanka;

lanka = **record**

x: integer;

y: integer;

next: list;

**end**;

**var**

h: text;

m, s, f, g, temp: list;

i, w: integer;

**procedure** Create\_list\_1(**var** p: list);

**var**

temp: list;

x: integer;

**begin**

assign(h, 'dev.txt');

reset(h);

**new**(p);

**new**(temp);

temp := p;

temp^.next := nil;

**while not** eof(h) **do**

**begin**

x := 0;

read(h, x);

**if** x = 0 **then begin** readln(h, x);**continue**; **end**;

temp^.x := x;

x := 0;

readln(h, x);

temp^.y := x;

**new**(temp^.next);

temp := temp^.next;

temp^.next := nil;

**end**;

close(h);

**end**;

**procedure** Create\_list\_2(**var** p: list);

**var**

temp: list;

x, y: integer;

**begin**

assign(h, 'dev.txt');

reset(h);

**new**(p);

**new**(temp);

temp := p;

temp^.next := nil;

**while not** eof(h) **do**

**begin**

read(h, x);

readln(h, y);

**if** (x = 0) **then begin continue**; **end**;

temp^.x := x \* y;

temp^.y := y - 1;

**new**(temp^.next);

temp := temp^.next;

temp^.next := nil;

**end**;

close(h);

**end**;

**procedure** Print\_list(**var** p: list);

**var**

temp: list;

**begin**

temp := p;

**while** temp^.next <> nil **do**

**begin**

write(temp^.x, 'x^', temp^.y);

writeln;

temp := temp^.next;

**end**;

writeln('.');

**end**;

**procedure** Delete\_list(**var** p: list);

**var**

temp: list;

**begin**

temp := p;

**while** p <> nil **do**

**begin**

p := temp^.next;

dispose(temp);

temp := p;

**end**;

**end**;

**function** Calc(p: list; a: integer): integer;

**var**

i, cal, sum: integer; s: integer;

**begin**

s := 1;

**while** p <> nil **do**

**begin**

**for** i := 1 **to** p^.y **do**

s := s \* a;

cal := p^.x \* s;

sum := sum + cal;

p := p^.next;

**end**;

calc := sum;

**end**;

**begin**

writeln('Vvedit znachenya x');

read(w);

Create\_list\_1(m);

Create\_list\_2(s);

f := m;

g := s;

write('Znachenya polinomy: ');

writeln( Calc(f, w));

write('Znachenya pohidnoi: ');

writeln( Calc(g, w));

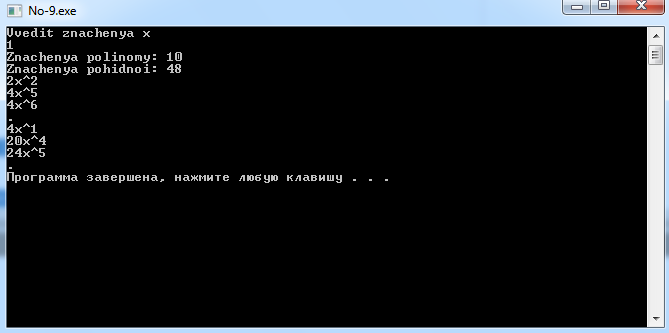
Print\_list(f);

Print\_list(g);

Delete\_list(f);

Delete\_list(g);

**end**.

1. Апробація програми. 

Завдання №10

1. Умова завдання.

Зобразити на екрані прямокутник, що обертається в площині екрану навколо свого центру.

1. Алгоритм розв’язання.

Крок 1.Виражаєм координати точки через тригонометричні перетворення, кожного разу додаючи певний кут.

Крок 2.Створюємо таких 4 точки.

Крок 3.Починаєм переміщати точки ,стираючи попередні.

1. Програмна реалізація.

**uses** GraphABC;

**var** i:integer;xv1,yv1,xv2,yv2,xv3,yv3, xv0,yv0, xc,yc,x2,y2,x1,y1, x0,y0,x3,y3:integer; gamma:real;

**procedure** rotate(x0,y0:integer;x1,y1:integer;gamma:real;**var** x2,y2:integer);

**var** r:real; beta,alpha:real;

**begin**

r:=sqrt(sqr(x1-x0)+sqr(y1-y0));

alpha:=arctan((x1-x0)/(y1-y0));

**if** (x1-x0)<0 **then**

alpha:=Pi-alpha;

beta:=gamma+alpha;

x2:=Round(r\*cos(beta))+x0;

y2:=Round(r\*sin(beta))+y0;

**end**;

**begin**

x1:=200;

y1:=100;

x2:=300;

y2:=300;

xc:=(x2+x1) **div** 2;

yc:=(y2+y1) **div** 2;

**for** i:=0 **to** 1000 **do**

**begin**

Clearwindow;

gamma:=(i\*Pi)/180;

rotate(xc,yc,x1,y1,gamma,xv0,yv0);

rotate(xc,yc,x2,y2,gamma,xv1,yv1);

rotate(xc,yc,x1,y2,gamma,xv2,yv2);

rotate(xc,yc,x2,y1,gamma,xv3,yv3);

Line(xv0,yv0,xv1,yv1);

Line(xv1,yv1,xv3,yv3);

Line(xv3,yv3,xv2,yv2);

Line(xv2,yv2,xv0,yv0);

sleep(50);

**end**;

**end**.

Завдання №12

1. Умова завдання.
2. Клас «підключення»: назва мережі, дата підключення (місяць і рік), вартість за місяць, наявність пільги, розмір пільги (у %).
3. Похідні: «за контрактом» (номер рахунку, дата відкриття, залишок), «за карткою» (номер рахунку, дата закінчення, залишок).
4. Описати конструктори, деструктори та необхідні методи. В усіх класах визначити функцію *print –* для кожного класу з різною реалізацією. Створити масив підключень і впорядкувати його за типом, назвою і залишком. Для тих, хто користується підключенням понад 5 років збільшити розмір пільги). Визначити підключення, в яких є достатньо грошей, щоб оплатити послугу за три наступні місяці; в яких картка не є дійсною вже два місяці, але сума для оплати за цей період на рахунку є.
5. Алгоритм розв’язання.

Крок 1.Створюємо батьківський клас і його дочірні.

Крок 2.В основній програмі реалізуємо переліченні вище завдання.

1. Програмна реалізація.

**Unit** Connection;

**interface**

**type**

Connection\_=**class**

**public**

name:string[20];

year:integer;

month:integer;

is\_pilga:integer;

rozmir\_pilgi:integer;

vartist:integer;

**Procedure** Read\_dani;

**Procedure** Connection\_.Copy(b:Connection\_);

**Procedure** Write\_dani;

**end**;

**implementation**

**Procedure** Connection\_.Read\_dani;

**begin**

write('Nazva meregi : ');

readln(name);

writeln(' Data pidcluchenya(month): ');

readln(month);

writeln(' Data pidcluchenya(year): ');

readln(year);

writeln(' Nayavnust pilgi(1-nayavna,2-vidcutnya): ');

readln(is\_pilga);

writeln(' Rozmir pilgi: ');

readln(rozmir\_pilgi);

writeln('Vartist za misatz:');

readln(vartist);

**end**;

**Procedure** Connection\_.Write\_dani;

**begin**

write('Nazva meregi : ');

writeln(name);

write(' Data pidcluchenya(month): ');

writeln(month);

write(' Data pidcluchenya(year): ');

writeln(year);

write(' Nayavnust pilgi: ');

writeln(is\_pilga);

write(' Rozmir pilgi: ');

writeln(rozmir\_pilgi);

writeln('Vartist za misatz:');

writeln(vartist);

**end**;

**Procedure** Connection\_.Copy(b:Connection\_);

**begin**

name:=b.name;

month:=b.month;

year:=b.year;

is\_pilga:=b.is\_pilga;

rozmir\_pilgi:=b.rozmir\_pilgi;

vartist:=b.vartist;

**end**;

**end**.

**Unit** For\_carta;

**interface**

**uses** Connection;

**type** For\_carta\_=**class**(Connection\_)

**public**

zalychok:real;

nomer\_rachynky :integer;

year:integer;

month:integer;

**Procedure** Read\_dani;

**Procedure** Write\_dani;

**end**;

**implementation**

**Procedure** For\_carta\_.Read\_dani;

**begin**

writeln(' Nomer rachynky:');

readln(nomer\_rachynky);

writeln(' Data zakinchenya(month): ');

readln(month);

writeln(' Data zakinchenya(year): ');

readln(year);

writeln(' Zalychok:');

readln(zalychok);

**end**;

**Procedure** For\_carta\_.Write\_dani;

**begin**

**inherited** Write\_dani;

write(' Nomer rachynky:');

writeln(nomer\_rachynky);

write(' Data zakinchenya(month): ');

writeln(month);

write(' Data zakinchenya(year): ');

writeln(year);

write(' Zalychok:');

writeln(zalychok);

**end**;

**end**.

**Unit** For\_contract;

**interface**

**uses** Connection;

**type**

For\_contract\_=**class**(Connection\_)

**public**

zalychok:real;

nomer\_rachynky :integer;

year:integer;

month:integer;

**Procedure** Read\_dani;

**Procedure** Write\_dani;

**end**;

**implementation**

**Procedure** For\_contract\_.Read\_dani;

**begin**

writeln(' Nomer rachynky:');

readln(nomer\_rachynky);

writeln(' Data vidkrutya(month): ');

readln(month);

writeln(' Data vidkrutya(year): ');

readln(year);

writeln(' Zalychok:');

readln(zalychok);

**end**;

**Procedure** For\_contract\_.Write\_dani;

**begin**

**inherited** Write\_dani;

write(' Nomer rachynky:');

writeln(nomer\_rachynky);

write(' Data vidkrutya(month): ');

writeln(month);

write(' Data vidkrutya(year): ');

writeln(year);

write(' Zalychok:');

writeln(zalychok);

**end**;

**end**.

**program** p;

**uses**

Connection, For\_carta, For\_contract;

**const**

n = 3;

**type**

mas = **array**[1..n]**of** Connection\_;

mass= **array**[1..n]**of** For\_carta\_;

masss= **array**[1..n]**of** For\_contract\_;

**var**

i: integer;

temp: Connection\_;

title: string[20];

k, l, t, sh: integer;

Connection1, Connection2, Connection3,Connection4: mas;

For\_carta1: mass;

For\_contract1: masss;

yea:integer;

mount:integer;

w:integer;

**procedure** Sort(**var** Connection2: mas; k: integer);

**var**

i, j: integer;

**begin**

**for** j := 1 **to** k - 1 **do**

**for** i := 1 **to** k - j **do**

**if**(Connection1[i].name > Connection1[i + 1].name) **then**

**begin**

temp := Connection1[i];

Connection2[i] := Connection2[i + 1];

Connection2[i + 1] := temp;

**end**;

**end**;

**procedure** Sort\_zalychok(**var** For\_carta1:mass);

**var** i,j:integer;max:real;

**begin**

max:=0;

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

**if** (For\_carta1[i].zalychok<For\_carta1[j].zalychok) **and** (j>i) **then**

**begin**

max:=For\_carta1[j].zalychok;

For\_carta1[j].zalychok:=For\_carta1[i].zalychok;

For\_carta1[i].zalychok:=max;

**end**;

**end**;

**procedure** Sort\_zalychok1(**var** For\_contract1:masss);

**var** i,j:integer;max:real;

**begin**

max:=0;

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

**if** (For\_contract1[i].zalychok<For\_contract1[j].zalychok) **and** (j>i) **then**

**begin**

max:=For\_contract1[j].zalychok;

For\_contract1[j].zalychok:=For\_contract1[i].zalychok;

For\_contract1[i].zalychok:=max;

**end**;

**end**;

**begin**

**for** i := 1 **to** n **do**

**begin**

Connection1[i] := Connection\_.Create;

Connection2[i] := Connection\_.Create;

Connection3[i] := Connection\_.Create;

Connection4[i] := Connection\_.Create;

Connection1[i].Read\_dani;

writeln('---------------------------');

**end**;

Sort(Connection2, k);

**for** i := 1 **to** n **do**

write('Pidcluchenya', i, ': ', Connection1[i].name, ' ');

writeln;

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

**for** i := 1 **to** n **do**

**begin**

For\_carta1[i] := For\_carta\_.Create;

For\_contract1[i] := For\_contract\_.Create;

**end**;

writeln;

k := 0;

**repeat**

writeln(' Vedit nazvy pidluchen,za contractom, else vedit (stop)');

readln(title);

**for** i := 1 **to** n **do**

**if** Connection1[i].name = title **then**

**begin**

k := k + 1;

writeln(' Vedit dani dlya contracty');

Connection2[k].Copy(Connection1[i]);

For\_contract1[k].Read\_dani;

Connection1[i] := Connection\_.Create;

**end**;

**until** (title = 'stop') **or** (k = n);

writeln('===========================================================');

L := k;

Sort(Connection2, L);

**for** k := 1 **to** L **do**

For\_contract1[k].Copy(Connection2[k]);

k := 0;

**repeat**

writeln(' Vedit nazvy pidluchen,za cartoy, else vedit (stop)');

readln(title);

**for** i := 1 **to** n **do**

**if** Connection1[i].name = title **then**

**begin**

k := k + 1;

writeln(' Vedit dani dlya carty');

Connection3[k].Copy(Connection1[i]);

For\_carta1[k].Read\_dani;

Connection1[i] := Connection\_.Create;

**end**;

**until**(title = 'stop') **or** (k = n) **or** (k + L = n);

writeln('===========================================================');

Sh := k;

Sort(Connection3, Sh);

**for** k := 1 **to** Sh **do**

For\_carta1[k].Copy(Connection3[k]);

k := 1;

**for** i := 1 **to** n **do**

**if** Connection1[i].name = '' **then**

**begin**

Connection1[i] := Connection2[k];

k := k + 1;

**end**;

k := 1;

**for** i := 1 **to** n **do**

**if** Connection1[i].name = '' **then**

**begin**

Connection1[i] := Connection3[k];

k := k + 1;

**end**;

writeln(' Вiдсортований масив:');

writeln('--------------------');

**for** i := 1 **to** n **do**

Connection1[i].Write\_dani;

writeln('=======================================================');

readln;

writeln(' Pidkluchenya za contractom:');

writeln('--------------------');

**for** i := 1 **to** L **do**

For\_contract1[i].Write\_dani;

writeln('=======================================================');

writeln(' Pidkluchenya za cartoy:');

writeln('------------------');

**for** i := 1 **to** Sh **do**

For\_carta1[i].Write\_dani;

writeln('===========================================================');

Sort\_zalychok(For\_carta1);

Sort\_zalychok1(For\_contract1);

writeln('Sort za zalychkom');

readln;

writeln(' Pidkluchenya za contractom:');

writeln('--------------------');

**for** i := 1 **to** L **do**

For\_contract1[i].Write\_dani;

writeln('=======================================================');

writeln(' Pidkluchenya za cartoy:');

writeln('------------------');

**for** i := 1 **to** Sh **do**

For\_carta1[i].Write\_dani;

writeln('===========================================================');

writeln('Vedit rik(na danuy chas)');

read(yea);

writeln('vedit misatz(na danuy chas');

read(mount);

writeln('Naskilki zbilchyty pilgy');

read(w);

**for** i:=1 **to** n **do**

**if** ((yea-Connection1[i].year)>5) **and** (mount>Connection1[i].month) **then**

Connection1[i].rozmir\_pilgi:=Connection1[i].rozmir\_pilgi+w;

**for** i:=1 **to** n **do**

**begin**

**if** For\_carta1[i].zalychok>=(For\_carta1[i].vartist)\*3 **then**

**begin**

writeln('Mozna oplatyty 3 misatza takomy pidkluchenuy');

For\_carta1[i].write\_dani;

**end**;

**if** For\_contract1[i].zalychok>=(For\_contract1[i].vartist)\*3 **then**

**begin**

writeln('Mozna oplatyty 3 misatza takomy pidkluchenuy');

For\_contract1[i].write\_dani;

**end**;

**if** For\_carta1[i].zalychok>=(For\_carta1[i].vartist) **then begin if** ((yea-For\_carta1[i].year)<=0)

**then begin if** (mount-For\_carta1[i].month<=-2) **then**

**begin**

writeln('Kartka ne diysna 2 misatsya ale syma do oplaty e');

For\_carta1[i].Write\_dani;

**end**;

**end**;

**end**;

**end**;

**end**.

Завдання №13

1. Умова завдання.

На прямій зафарбувати **n** відрізків. Відрізки задаються координатами початку та кінця. Знайти загальну довжину зафарбованих частин прямої.

1. Алгоритм розв’язання.

Крок 1.Визначаємо взаємне розташування відрізків (накладаються чи ні).

Крок 2.Оббчислюємо довжинну.

1. Програмна реалізація.

unit Unit1;  
  
interface  
  
uses  
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ExtCtrls;  
  
type  
TForm1 = class(TForm)  
PaintBox1: TPaintBox;  
Edit1: TEdit;  
Button1: TButton;  
Button2: TButton;  
procedure PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;  
Shift: TShiftState; X, Y: Integer);  
procedure Button1Click(Sender: TObject);  
procedure Button2Click(Sender: TObject);  
procedure FormCreate(Sender: TObject);  
private  
{ Private declarations }  
public  
{ Public declarations }  
end;  
  
var  
Form1: TForm1;  
  
implementation  
  
{$R \*.dfm}  
var a:array[1..2]of TPoint;  
i,xc,yc:integer;  
procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button: TMouseButton;  
Shift: TShiftState; X, Y: Integer);  
begin  
i:=i+1;  
paintBox1.Canvas.Pixels[x,y]:=clRed;  
a[i].X:=x;  
a[i].Y:=y;   
end;  
  
procedure TForm1.Button1Click(Sender: TObject);  
begin  
paintBox1.Canvas.Pen.Color:=clSilver;  
paintBox1.Canvas.Pen.Style:=psDot;  
paintBox1.Canvas.MoveTo(a[1].x,a[1].y);  
paintBox1.Canvas.LineTo(a[2].x,a[2].y);  
xc:=a[1].X;  
yc:=a[1].Y;  
end;  
  
procedure TForm1.Button2Click(Sender: TObject);  
var x,y,k,h:integer;  
begin  
x:=strtoint(edit1.Text);  
k:=abs(a[2].Y-a[1].Y);  
h:=trunc(sqrt(sqr(a[2].X-a[1].X)+sqr(a[2].Y-a[1].Y)));  
y:=trunc((k\*abs(x-xc)/(sqrt(h\*h-k\*k))+yc));  
paintBox1.Canvas.Pen.Color:=RGB(Random(256), Random(256), Random(256));  
paintBox1.Canvas.Pen.Width:=3;  
paintBox1.Canvas.MoveTo(xc,yc);  
paintBox1.Canvas.LineTo(x,y);  
xc:=x;  
yc:=y;  
end;  
  
procedure TForm1.FormCreate(Sender: TObject);  
begin  
Form1.Color:=clWhite;  
  
end;  
  
end.

1. Апробація програми.

