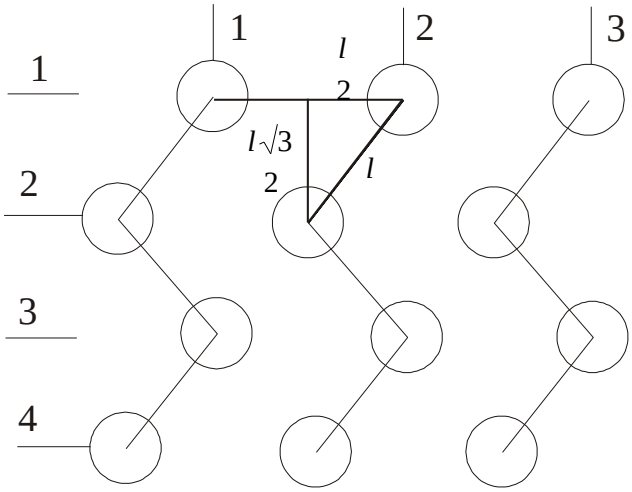
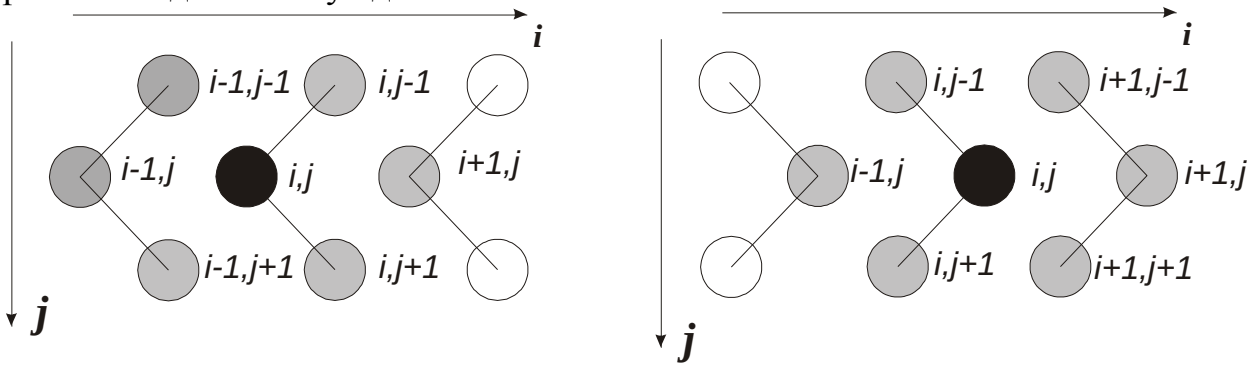
Модель двовимірної упорядкованої структури

Оскільки квадратна гратка є нестійкою для однокомпонентної системи, то опишемо трикутну гратку, яка має шість сусідів у першій координаційній сфері:



Нумерація вузлів у двовимірній трикутній гратці

Така нумерація дозволяє уникнути пустих елементів масиву при описі частинок у вигляді двовимірного масиву. Але виникає два види вузлів з різними індексами сусідів:



j парне j непарне

Індекси сусідів для вузлів трикутної гратки у випадку парних та непарних рядків

Парність рядка можна визначити через функцію залишку від цілочисельного ділення. Тому процедуру *an* визначення сили парної взаємодії атома з сусідом необхідно виконати для кожного сусіда в залежності від парності/непарності номера стовпця атома:

```
if j mod 2=0
  then
     begin
          an (i-1,j-1)
          an (i,j-1)
          an (i+1,j)
          an (i,j+1)
          an (i-1,j+1)
          an (i-1,j)
      end
```

```
else
     begin
          an (i,j-1)
          an (i+1,j-1)
          an (i+1,j)
          an (i,j+1)
          an (i-1,j+1)
          an (i-1,j)
     end;
```

Для спрощення можна використати масив зміщень індексів, де нульовий та перший рядок відповідають зміщенням по стовпчиках в залежності від парності рядка, а другий рядок відповідає зміщенням по рядках:

Neig:array [1..3,1..6] of byte = $\begin{pmatrix} -1 & 0 & -1 & +1 & -1 & 0 \\ +1 & 0 & -1 & +1 & +1 & 0 \\ -1 & -1 & 0 & 0 & +1 & +1 \end{pmatrix} \begin{matrix} //\Delta i,\ if\ \ j \bmod 2 = 0 \\ //\Delta i,\ if\ \ j \bmod 2 = 1 \\ //\Delta j \end{matrix}$

У результаті описаний вище фрагмент визначення сил парної взаємодії атома *(i,j)* з сусідом an можна викликати у циклі по номерах сусідів (numb_neig=6) додаткової перевірки парності/непарності номера стовпчика:

```
for k:=1 to numb_neig do
  an(i+neig[j mod 2,k],j+neig[2,k])
```

Приклад використання періодичної упорядкованої структури при дослідженні стабільних конфігурацій методом молекулярної статики

Програма складається з таких підпрограм:

- ініціалізація початкових умов;

```
procedure init;                  //розміщення атомів в рівноважних вузлах
begin
 Form1.Chart1.Height:=round(Form1.Chart1.Width*sqrt(3)/2);
 Form1.Chart1.LeftAxis.Maximum:=(n+1)*a*sqrt(3)/2;
 Form1.Chart1.LeftAxis.Minimum:=0;
 Form1.Chart1.BottomAxis.Maximum:=(n+1)*a+a/2;
 Form1.Chart1.BottomAxis.Minimum:=0;
 for i:=1 to n do
  for j:=1 to n do
   begin
    x[i,j]:=a*i+(j mod 2)*a/2;
    y[i,j]:=a*j*sqrt(3)/2;
    color[i,j]:=1;
  // if (i>=5) and (i<=7) then color[i,j]:=0
   end;
 color[5,6]:=0;
 for i:=1 to 6 do color[5+neig[6 mod 2,i],6+neig[2,i]]:=0;
 visual
end;
```

- періодичні граничні умови Борна Кармана;

```
procedure bk(var ia,ja:integer);
 begin
  dx:=0;dy:=0;
  if ia<1 then
   begin ia:=n+ia; dx:=-n*a end;
  if ia>n then
   begin ia:=ia-n; dx:=n*a  end;
  if ja<1   then
   begin ja:=n+ja; dy:=-n*a*sqrt(3)/2 end;
  if ja>n then
   begin ja:=ja-n; dy:=n*a*sqrt(3)/2  end;
 end{bk};
```

- підпрограми перебору сусідів заданого атома в межах першої координаційної сфери;

```
procedure atom_neighbours(ian,jan:integer);
 var temp:longint;
    r:double;
```

- функція для визначення потенціальної енергії парної взаємодії;

```
function p(r:double; col1,col2:integer):double;
 begin
  if col1*col2<>0
   then p:=4*sigma*(exp(12*ln(r0/r))-exp(6*ln(r0/r)))
   else p:=0
 end;
```

- функція для визначення сили енергії парної взаємодії;

```
function F(r:double; col1,col2:integer):double;
 begin
  if col1*col2<>0
   then F:=24*sigma/r*(2*exp(12*ln(r0/r))-exp(6*ln(r0/r)))
   else f:=0
 end{bmp};                                   //Сатон-Чена
```

- визначення проекцій сили;

```
procedure an(iss,jss:integer);
 var ff:double;
 begin
  BK(iss,jss);
  r:=sqrt(sqr(x[ian,jan]-(x[iss,jss]+dx))+sqr(y[ian,jan]-(y[iss,jss]+dy)));
  ff:=F(r,color[ian,jan],color[iss,jss]);
  Fx[ian,jan]:=Fx[ian,jan]+ff*(x[ian,jan]-(x[iss,jss]+dx))/r;
  Fy[ian,jan]:=Fy[ian,jan]+ff*(y[ian,jan]-(y[iss,jss]+dy))/r;
  p_ene:=p_ene+p(r,color[ian,jan],color[iss,jss]);
 end{an};
begin
 for i:=1 to numb_neig do
  an(ian+neig[jan mod 2,i],jan+neig[2,i])
end{atom_neighbours};
```

- визначення кроку по часу *dt* (через пошук максимальної проекції сили після ініціалізації);

```
function found_dt:real;
var f_max:real;
i,j: integer;
 begin
   f_max:=0;
    for i:=1 to n do
     for j:=1 to n do
      if color[i,j]<>0 then
       begin
        Fx[i,j]:=0;
        Fy[i,j]:=0;
       end;
    for i:=1 to n do
     for j:=1 to n do
      if color[i,j]<>0 then atom_neighbours(i,j);
    for i:=1 to n do
     for j:=1 to n do
       if color[i,j]<>0 then
        begin
         if f_max<abs(Fx[i,j]) then f_max:=abs(Fx[i,j]);
         if f_max<abs(Fy[i,j]) then f_max:=abs(Fy[i,j]);
        end;
  found_dt:=f_max
 end;
```

- алгоритм методу молекулярної статики;

```
procedure MS;
var i,j: integer;
    f_max:double;
 begin
    p_ene_old:=p_ene;
    p_ene:=0;
    f_max:=0;
    for i:=1 to n do
     for j:=1 to n do
      if color[i,j]<>0 then
       begin
        Fx[i,j]:=0;
        Fy[i,j]:=0;
       end;
    for i:=1 to n do
     for j:=1 to n do
      if color[i,j]<>0 then atom_neighbours(i,j);
    for i:=1 to n do
     for j:=1 to n do
       if color[i,j]<>0 then
        begin
         if f_max<abs(Fx[i,j]) then f_max:=abs(Fx[i,j]);
         if f_max<abs(Fy[i,j]) then f_max:=abs(Fy[i,j]);
        end;
    for i:=1 to n do
     for j:=1 to n do
       begin
        if color[i,j]<>0 then
         begin
          x[i,j]:=x[i,j]+Fx[i,j]/2/massa*sqr(dt);
          y[i,j]:=y[i,j]+Fy[i,j]/2/massa*sqr(dt);
         end;
       end;
   Form1.Series2.AddXY(t/dt,f_max*1e11);
   Form1.Series3.AddXY(t/dt,p_ene*1e20);
   t:=t+dt;
   visual;
   Application.ProcessMessages
end;
```

- візуалізація;

```
procedure visual;
begin
 Form1.Series1.Clear;
 for i:=1 to n do
  for j:=1 to n do
   begin
    case color[i,j] of
     0: Form1.Series1.AddBubble(x[i,j],y[i,j],0.45*r0,",clBtnFace);
     1: Form1.Series1.AddBubble(x[i,j],y[i,j],0.45*r0,",clred);
     2: Form1.Series1.AddBubble(x[i,j],y[i,j],0.45*r0,",clgreen);
    end;
   end;
 Application.ProcessMessages
end;
```

- головна програма;

```
init;
 dt:=found_dt;
 p_ene_old:=0;p_ene:=-1e-17;
 while stop and (abs(p_ene_old-p_ene)>1e-35) do
  begin
   ms;
  end;
```