II Етап

Завдання:
1. Додати до проекту малювання ребер.
2. Створення та редагування графа:
- зображення пронумерованих вершин
- зображення ребер
- можливість видалення ребер
- можливість видалення вершин
3. Запис графа у файл на диск через меню File-Save.
4. Зчитування графа з диска через меню File-Open.
5. Завершення роботи програми через меню File-Exit.

Хід виконання:

1. Для малювання ребер потрібно буде опрацьовувати події натискання кнопки мишки, переміщення курсору мишки і відпускання кнопки мишки. Для зберігання інформації про існуючі ребра потрібно описати матрицю суміжності, в якій буде зберігатись номери обох вершин для кожного ребра. Додамо відповідний тип даних:

```
  //матриця
  TMATRIX = array[1..CMAS_MAXSIZE,1..2] of integer;
```

І додаткові змінні для граней:

```
  //кількість граней
  EdgesCount: integer;
  //матриця суміжності двох вешнин для грані
  edges: TMATRIX;
  //координати початку і кінця для грані
  startnode, endnode: Integer;
  //координати поточної позиції мишки при малювані грані
  tmppoint : TPoint;
```

Також необхідно внести зміни для ініціалізації початкових даних:

```
procedure TfrmMain.PrepareData;
var
  i, j: integer;
begin
  //ініціалізація змінних
  NodesCount := 0;
  EdgesCount := 0;
  for i := 1 to CMAS_MAXSIZE do
  begin
    mas_x[i] := 0;
    mas_y[i] := 0;
    mas_n[i] := 0;
    for j:=1 to CMAS_MAXSIZE do edges[i,j] := 0;
  end;
end;
```

Непогано буде винести код пошуку вершини по координатах теж окремою функцією, яка буде повертати № вершини в масиві або -1 якщо координати не відносяться до жодної з існуючих вершин:

```
function TfrmMain.FindNodeByXY(x, y: integer): integer;
var
  i: Integer;
begin
  Result := -1;
  if NodesCount = 0 then Exit;
  for i := 1 to NodesCount do
```

```
  begin
    if (X > (MAS_X[i] - CNODE_RADIUS)) and (X < (MAS_X[i] + CNODE_RADIUS)) and
      (Y > (MAS_Y[i] - CNODE_RADIUS)) and (Y < (MAS_Y[i] + CNODE_RADIUS)) then
    begin
      Result := i;
      Exit;
    end;
  end;
end;
```

З врахуванням цієї функції внесемо зміни в метод OnMouseDown():

```
  p: Integer; //№ вершини

begin
  //режим додавання вершин
  if (SpeedButton1.Down = True) then
  begin
    //якщо натиснута ліва кнопка мишки
    if (ssLeft in Shift) then
    begin
      p := FindNodeByXY(x, y);
      //додаємо нову вершину
      if p=-1 then
      begin
        NodesCount := NodesCount + 1;
        MAS_X[NodesCount] := X;
        MAS_Y[NodesCount] := Y;
        MAS_N[NodesCount] := NodesCount;
        draw_graph_node(X, Y, NodesCount);
      end
      else
      begin
        //такі координати вже зайняті - вершину не додаємо
        Memo1.Lines.Add(
          'В цьому місці вже існує вершина!');
        Memo1.CaretPos := Point(0, Memo1.Lines.Count - 1);
      end;
    end else if (ssRight in Shift) then begin
     //todo: видалення вершини правою кнопкою мишки
    end;
  end;
```

Аналогічно напишемо функцію для пошуку ребра по номерах вершин:

```
function TfrmMain.FindEdgeByNodes(start, finish: Integer): Integer;
var
  i: Integer;
begin
  Result:=-1;
  if EdgesCount=0 then Exit;
  for i:=1 to EdgesCount do begin
   //при пошуку потрібно враховувати, що початкова і кінцева вершини можуть
бути поміняні місцями
   if ((edges[i,1]=start) and (edges[i,2]=finish)) or ((edges[i,2]=start) and
(edges[i,1]=finish))
     then begin
       Result:=i;
       Exit;
     end;
  end;
end;
```

Процес малювання графа тепер краще винести окремою процедурою:

```
procedure TfrmMain.DrawGraph;
var
  i: Integer;
begin
  if NodesCount=0 then Exit;
  for i:=1 to NodesCount do draw_graph_node(mas_x[i], mas_y[i], mas_n[i]);
  if EdgesCount=0 then Exit;
  for i:=1 to EdgesCount do begin
    //малюємо лінію від обраної вершири до порточних координат
    Image1.Canvas.Pen.Style := psSolid;
    Image1.Canvas.MoveTo(mas_x[edges[i,1]],mas_y[edges[i,1]]);
    Image1.Canvas.LineTo(mas_x[edges[i,2]],mas_y[edges[i,2]]);
  end;
  Invalidate;
end;
```

Після попередньої підготовки можна додати сам процес малювання граней, допрацюємо спочатку метод OnMouseDown():

```
 //режим додавання ребер
  if (SpeedButton2.Down = True) then
  begin
    //якщо менше двох вершин, не можна додати ребро
    if NodesCount < 2 then
    begin
      Memo1.Lines.Add('Недостатньо вершин, щоб додати грань!');
      Memo1.CaretPos := Point(0, Memo1.Lines.Count - 1);
      Exit;
    end;
    //шукаємо координати обраної вершини
    startnode:=FindNodeByXY(x,y);
    endnode:=-1;
  end;
```

Тепер додамо метод OnMouseMove():

```
procedure TfrmMain.Image1MouseMove(Sender: TObject; Shift: TShiftState; X, Y:
integer);
begin
  //якщо не обрано режим малювання ребер або недостатньо вершин, нічого не
робимо
  if (NodesCount<2) or (SpeedButton1.Down)or (startnode=-1) then Exit;

  ClearImage;
  DrawGraph;

  //малюємо лінію від обраної вершири до порточних координат
  Image1.Canvas.Pen.Style := psSolid;
  Image1.Canvas.Pen.Color:=clRed;
  Image1.Canvas.MoveTo(mas_x[startnode],mas_y[startnode]);
  Image1.Canvas.LineTo(x,y);
  Invalidate;
end;
```

І нарешті метод OnMouseUp(), в якому буде додаватись намальоване ребро в матрицю:

```
procedure TfrmMain.Image1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
```

```
  newedge : Integer;
begin
  //не режим додавння ребер - нічого не робимо
  if SpeedButton1.Down then Exit;
  endnode:=FindNodeByXY(x,y);
  if endnode=-1 then begin
    Memo1.Lines.Add('Не обрано кінцеву вершину!');
    Memo1.CaretPos := Point(0, Memo1.Lines.Count - 1);
    Exit;
  end;
  newedge:=FindEdgeByNodes(startnode,endnode);
  if newedge=-1 then begin
   EdgesCount:=EdgesCount+1;
   edges[EdgesCount,1]:=startnode;
   edges[EdgesCount,2]:=endnode;
  end;
end;
```

Тепер додамо можливість видалення ребер. Будемо це робити аналогічно малюванню, але правою кнопкою мишки. Додамо що одну змінну, яка буде визначати режим додавання або видалення:

```
  //ознака режиму видалення
  deletemode : Boolean;
```

Будемо встановлювати цю ознаку в методі OnMouseDown():

```
    //якщо натиснуто праву кнопку мишки, режим видалення ребра
    if (ssRight in Shift) then deletemode:=True
    else if (ssLeft in Shift) then deletemode:=False;
```

Тепер потрібно додати функцію, яка буде видаляти ребро з матриці по його номеру і зміщувати всі наступні елементи:

```
procedure TfrmMain.RemoveEdge(num: integer);
var
  i: integer;
begin
  if num = -1 then Exit;
  if num < EdgesCount then
  begin
    for i := num to EdgesCount - 1 do
    begin
      edges[i, 1] := edges[i + 1, 1];
      edges[i, 2] := edges[i + 1, 2];
    end;
  end;
  edges[EdgesCount, 1] := 0;
  edges[EdgesCount, 2] := 0;
  EdgesCount := EdgesCount - 1;
end;
```

Залишилось доробити видалення вершини. Найкраще для цього підійде рекурсивна процедура, яка буде видаляти усі ребра, пов'язані з цією вершиною.

```
procedure TfrmMain.RemoveNode(nodenum: integer);
var
  found: boolean;
  //ознака знайденого ребра з потрібною вершиною
  i, j: integer;
begin
  found := False;
  for i := 1 to EdgesCount do
```

```
  begin
    if (edges[i, 1] = nodenum) or (edges[i, 2] = nodenum) then
    begin
      //якщо було знайдено ребро з зазначеною вершиною, видаляємо його
      //і перериваємо цикл зі встановленням ознаки
      RemoveEdge(i);
      Found := True;
      Break;
    end;
  end;
  //якщо ознака встановлена, виконуємо процедуру рекурсивно
  if found then begin
    RemoveNode(nodenum);
    Exit;
  end
  else
  begin
    //ознака не встановлена - вершина більше не прив'язана до жодного з ребер,
    //отже її можна видалити (по аналогії видалення ребра - зміщуємо елементи
масиву)
    if nodenum < NodesCount then
    begin
      for j := nodenum to NodesCount - 1 do
      begin
        mas_x[j] := mas_x[j + 1];
        mas_y[j] := mas_y[j + 1];
        mas_n[j] := j;
      end;
    end;
    //очищаємо останній елемент масивів і зменшуємо кількість вершин
    mas_x[NodesCount] := 0;
    mas_y[NodesCount] := 0;
    mas_n[NodesCount] := 0;
    NodesCount := NodesCount - 1;
    //поправка на поточну кількість вершин
    if EdgesCount>0 then for j:=1 to EdgesCount do begin
      if edges[j,1]>=NodesCount then edges[j,1]:=edges[j,1]-1;
      if edges[j,2]>=NodesCount then edges[j,2]:=edges[j,2]-1;
    end;
  end;
end;
```

Тепер додаємо до програми меню і створюємо відповідні пункти. Крім цього, додамо з вкладки Dialogs палітри компонентів SaveDialog1 і OpenDialog1, щоб було зручно обирати файли для збереження/завантаження.
Для пункту File/Save напишемо код збереження інформації про граф в текстовому форматі. Структура файлу буде наступною (кожне число записане окремим рядком):
☐ в першому рядку буде вказуватись кількість вершин N
☐ Наступні рядки (N * 2) - координати X/Y кожної вершини, кожне число окремим рядком
☐ Після цього вказано кількість ребер E
☐ Якщо E=0, наступні рядки відсутні, інакше по кількості E*2 значення вершин, які об'єднуються ребром, аналогічно до координат X/Y вершин.
Ось код, який відповідає за запис файла:

```
procedure TfrmMain.MenuItem2Click(Sender: TObject);
var
  f : TextFile;
  i: Integer;
begin
  //Зберегти файл з графом
  if NodesCount=0 then Exit; //Якщо немає жодної вершини, немає що зберігати
  SaveDialog1.InitialDir:=ExtractFileDir(Application.ExeName);
  if SaveDialog1.Execute then begin
```

```
   AssignFile(f,SaveDialog1.FileName);
   Rewrite(f);
   WriteLn(f,NodesCount);
   for i:=1 to NodesCount do begin
     WriteLN(f,mas_x[i]);
     WriteLN(f,mas_y[i]);
   end;
   WriteLn(f,EdgesCount);
   if EdgesCount<>0 then for i := 1 to EdgesCount do begin
    WriteLn(f,edges[i,1]);
    WriteLn(f,edges[i,2]);
   end;
   CloseFile(f);
  end;
end;
```

По аналогії читаємо дані графа з файлу:

```
procedure TfrmMain.MenuItem3Click(Sender: TObject);
var
  f : TextFile;
  i: Integer;
  s : String;
begin
  //Прочитати файл з графом, очистивши попередню інформацію
  PrepareData;
  ClearImage;
  OpenDialog1.InitialDir:=ExtractFileDir(Application.ExeName);
  if OpenDialog1.Execute then begin
   AssignFile(f,OpenDialog1.FileName);
   Reset(f);
   ReadLN(f,s);
   NodesCount:=StrToInt(s);
   for i:=1 to NodesCount do begin
     ReadLN(f,s);
     mas_x[i]:=StrToInt(s);
     ReadLN(f,s);
     mas_y[i]:=StrToInt(s);
   end;
   ReadLN(f,s);
   EdgesCount:=StrToInt(s);
   if EdgesCount<>0 then for i := 1 to EdgesCount do begin
     ReadLN(f,s);
     edges[i,1]:=StrToInt(s);
     ReadLN(f,s);
     edges[i,2]:=StrToInt(s);
   end;
   CloseFile(f);
  end;
  ClearImage;
  DrawGraph;
end;
```

Для виходу з програми через меню File/Exit прописуємо наступний рядок:

```
Close;
```

Як бонус, через меню Help буде відображатись інформація про програму:

```
ShowMessage('Графобудівник - програма для візуальної побудови графів.');
```

І ще один бонус - збереження зображення графа в файл, який можна переглянути будь-якою програмою

для перегляду зображень.
Для цього на формі розміщено ще один діалог - SavePictureDialog1. Відповідний пункт меню File/Export image має наступний код:

```
procedure TfrmMain.MenuItem6Click(Sender: TObject);
begin
  SavePictureDialog1.InitialDir:=ExtractFileDir(Application.ExeName);
  if SavePictureDialog1.Execute then begin
   Image1.Picture.SaveToFile(SavePictureDialog1.FileName);
  end;
end;
```