```
IsDistinctQ[P_, q_] := Block[
                       программный блок
  {flag = True, i},
          истина
  For[i = 1, i ≤ Length[P], i++, {
 цикл ДЛЯ          длина
    If[q == P⟦i⟧, flag = False; Break[]
    условный оператор      ложь    прервать цикл
    ]}];
  Return[flag];
  вернуть управление
 ]

CreatePoints[Size_] := Block[
                       программный блок
  {K = {}, q, rang = 50, i},
  For[i = 1, i ≤ Size, i++, {
 цикл ДЛЯ
    q = {RandomInteger[{-rang, rang}], RandomInteger[{-rang, rang}]};
        случайное целое число                 случайное целое число
    If[IsDistinctQ[K, q], AppendTo[K, q], i--];
    условный оператор          добавить в конец к
   }];
  Return[K];
  вернуть управление
 ]

IsLeftQ[p1_, p2_, p3_] := Block[
                          программный блок
  {flag = True},
          истина
  If[Det[{p2 - p1, p3 - p1}] ≥ 0, flag = True, flag = False];
 ⋯ детерминант                          истина       ложь
  Return[flag];
  вернуть управление
 ]

FindStartLine[P_] := Block[
                     программный блок
  {first, second, i, minAngle = 2 π},
  first = FindMostDownPoint[P];
  For[i = 1, i ≤ Length[P], i++, {
 цикл ДЛЯ          длина
    If[VectorAngle[P⟦i⟧ - first, {1, 0}] < minAngle && P⟦i⟧ ≠ first, {
   ⋯ угол между векторами
      minAngle = VectorAngle[P⟦i⟧ - first, {1, 0}];
                 угол между векторами
      second = P⟦i⟧}]
   }];
  Return[{first, second}]
  вернуть управление
 ]

FindMostDownPoint[P_] := Block[
                         программный блок
```

```
                                    программный блок
  {i, mostDown = P[[1]]},
  For[i = 2, i ≤ Length[P], i++, {
  цикл ДЛЯ          длина
    If[mostDown[[2]] > P[[i]][[2]], mostDown = P[[i]]]
    условный оператор
   }];
  Return[mostDown]
  вернуть управление
 ]

FindMostBigAngle[P_, q1_, q2_] := Block[
                                  программный блок
  {i, angle = 0, res = {}},
  For[i = 1, i ≤ Length[P], i++, {
  цикл ДЛЯ          длина
    If[VectorAngle[q1 - P[[i]], q2 - P[[i]]] > angle && IsLeftQ[q1, q2, P[[i]]],
    … угол между векторами
     angle = VectorAngle[q1 - P[[i]], q2 - P[[i]]]; res = P[[i]]
             угол между векторами
   }];
  Return[res]
  вернуть управление
 ]

IsNewTriangular[Trian_] := Block[
                           программный блок
  {i, flag = True, STrian = Sort[Trian, #1[[1]] < #2[[1]] &]},
             истина          сортировать
  (*Print["STrian: ",STrian];*)
    печатать
  For[i = 1, i ≤ Length[Triangulation], i++, {
  цикл ДЛЯ          длина
    If[Sort[Triangulation[[i]], #1[[1]] < #2[[1]] &] == STrian, flag = False;
    … сортировать                                                  ложь
     Break[]]
     прервать цикл
   }];
  Return[flag]
  вернуть управление
 ]

CreateTriangulation[P_, q1_, q2_] := Block[
                                     программный блок
  {newPoint, i},

  newPoint = FindMostBigAngle[P, q1, q2];
  If[Length[newPoint] == 0, Return[], {
  … длина                   вернуть управление
    If[IsNewTriangular[{q1, newPoint, q2}], {
    условный оператор
      AppendTo[Triangulation, {q1, newPoint, q2}];
      добавить в конец к
      CreateTriangulation[P, q1, newPoint];
      CreateTriangulation[P, newPoint, q2];
```

```
      }]
    }];
 ]

Triangulation = {};
P = CreatePoints[100];
startLine = FindStartLine[P];
CreateTriangulation[P, startLine[[1]], startLine[[2]];

Graphics[{
 графика
   Table[Line[Append[Triangulation[[i]], Triangulation[[i]][[1]]]],
    табл··· (ло··· добавить в конец
     {i, 1, Length[Triangulation]}]
            длина
 }]
```