A solution to a nice programming problem.

The problem to be treated has served as a problem at a written examination some time ago. The solution to it is so nice, that it is worth to be recorded.

*

Given are two integer arrays x,y(0:46). The pair (x(i),y(i)) represents the Cartesian coordinates of a point P_i in a plane ($0 \leq i < 47$). The arrays are such that all 47 points are different.

A robot walks from P_0 to P_1, from P_1 to P_2, ..., from P_{45} to P_{46}, and finally from P_{46} to P_0. It does so in obedience to the following rules:

- it shall start from P_0, whilst looking towards P_1;
- it shall always walk into the direction in which it is looking;
- it shall only alter its direction of looking in the points P_i, namely by performing a clockwise rotation of α: $0 \leq \alpha < 2\pi$;
- it shall end in P_0, whilst looking towards P_1.

As a consequence of this walk the robot has made a clockwise rotation which is a multiple of 2π.

Write a program to compute this multiple, under the additional requirement that all expressions and variables occurring in the program have to be of integer or of boolean type.

A first inspection of the problem tells us that the requested answer does not change whenever the set of points is translated through the plane. Only the relative position of each pair of successive points plays a role in the determination of the answer.
Therefore we introduce the vectors $\underline{u}_i$, to be defined as $\underline{u}_i = (P_i \longrightarrow P_{i+1})$ for $0 \leq i \leq 47$, where $P_{47} = P_0$ and $P_{48} = P_1$.
The direction of looking during the walk from P_i to P_{i+1} then equals $\underline{u}_i$.
Hence the successive directions of looking during the robot's walk are
$\underline{u}_0, \underline{u}_1, \underline{u}_2, \cdots, \underline{u}_{46}, \underline{u}_{47}$ $(= \underline{u}_0)$.
The clockwise rotation in the point P_i towards P_{i+1} now corresponds to a clockwise rotation from $\underline{u}_{i-1}$ to $\underline{u}_i$.

Since in the initial state, the robot looks into the same direction as in the final state, it is suggested to take this directon as "reference-direction". And then the final answer will approximately be equal to total number of times that during rotation the robot's eye "passes" this reference-direction.
A more precise, and a more manageable formulation of the problem is at hand as soon as we realise that the answer does not change either if the set of points is rotated inside the plane. This has as a consequence that any direction $\underline{r}$ can be taken as a reference-direction.
If we define $\theta(\underline{u}_i) : 0 \leq \theta(\underline{u}_i) < 2\pi$ to be the angle of the clockwise rotation from $\underline{r}$ to $\underline{u}_i$, then the final answer p will have to satisfy the equation

$$R: \quad p = (\underline{N}i: 0 \leq i < 47: \ \theta(\underline{u}_{i+1}) < \theta(\underline{u}_i)) \ .$$

When taking as an invariant relation

$$Q_0: \quad p = (\underline{N}i: 0 \leq i < j: \ \theta(\underline{u}_{i+1}) < \theta(\underline{u}_i)) \ \underline{\text{and}} \ 0 \leq j \leq 47 \ ,$$

then the following program comes quite naturally:

```
j,p := 0,0;
do j ≠ 47 → if θ(u_{j+1}) < θ(u_j) → p := p + 1
            □ θ(u_{j+1}) ≥ θ(u_j) → skip
            fi;
            j := j + 1
od .
```

If the computation of θ is felt to be expensive, we can introduce

$$Q_1: \quad \alpha = \theta(\underline{u}_j)$$

and take as a strengthened invariant Q_0 and Q_1, giving rise to the program

```
j,p := 0,0; α := θ(u_0);
do j ≠ 47 → j := j + 1; β := θ(u_j);
            if β < α → p := p + 1
            □ β ≥ α → skip
            fi;
            α := β
od .
```

A next remark is that, in fact, we do not want to compute the angles α and β, but just to compare them.
Comparing two angles α and β is fairly easy in the case that $0 \leq \alpha < \pi/2$ and $0 \leq \beta < \pi/2$, because then $\beta < \alpha$ is equivalent to $\tan(\beta) < \tan(\alpha)$, tangents being compared at the expense of two simple multiplications, when points are given in a Cartesian coordinate system.
Comparing two angles α and β is not so difficult either in the general case, if we are willing to represent α and β in the angle system with base $\pi/2$:

$\alpha = a.\pi/2 + \alpha' \quad (0 \leq \alpha' < \pi/2)$

$\beta = b.\pi/2 + \beta' \quad (0 \leq \beta' < \pi/2)$.

Then, the inequality $\beta < \alpha$ is equivalent to

$b < a$ **or** $(b = a$ **and** $\beta' < \alpha')$,

which in turn is equivalent to

$b < a$ **or** $(b = a$ **and** $\tan(\beta') < \tan(\alpha'))$.

Now the program can, with some liberal notations, be rewritten as:

```
j.p := 0,0;  a,α' :( a.π/2 + α' = θ(u0)  and  0 ≤ α' < π/2 );
do j ≠ 47 → j := j + 1;
            b,β' :( b.π/2 + β' = θ(uj)  and  0 ≤ β' < π/2 );
            if b < a  or  (b = a  and  tan(β') < tan(α') → p := p + 1
             [] b > a  or  (b = a  and  tan(β') ≥ tan(α') → skip
            fi;
            a,α' := b,β'
od.
```

Looking for a simple initialisation of a and α', it seems attractive to choose $\underline{r} = \underline{u}_0$ as a reference-direction, but in that case the computations of b, β' and of $\tan(\beta') < \tan(\alpha')$ become rather cumbersome. With respect to these latter computations there is hardly another possibility than taking one of the four coordinate-directions as reference-direction $\underline{r}$. The choice being immaterial (on account of symmetry), we shall select the negative y-axis.

Now we switch back in our representation from angles to vectors, because in order to compute $\tan(\alpha')$ and $\tan(\beta')$ it is convenient to have two vectors (ux,uy) and (vx,vy) respectively, such that

$\theta(ux,uy) = \alpha'$ and $\theta(vx,vy) = \beta'$.
Then $\tan(\beta') < \tan(\alpha')$ is coded as $(-vx)/(-vy) < (-ux)/(-uy)$, or preferably as $vx*uy < ux*vy$.
In order to achieve that $0 \le \beta' < \pi/2$, we have to achieve that $vx \le 0$ and $vy < 0$. In order to compute b we have to investigate counterclockwise turns of $\pi/2$ of the vector $\underline{u}_j$.
Now we obtain for b,β' :($b.\pi/2 + \beta' = \theta(\underline{u}_j)$ and $0 \le \beta' < \pi/2$) the following piece of code:

```
vx,vy := x(j+1) - x(j),y(j+1) - y(j);  b := 0;
do vx > 0  or  vy ≥ 0  →  vx,vy := -vy,vx;  b := b + 1  od,
```

where, in view of the original data, $x(j+1) - x(j)$ and $y(j+1) - y(j)$ are the components of the vector $\underline{u}_j$.

The initialisation of a and α' can become quite simple if we succeed in finding some vector $\underline{u}_{-1}$ such that $\theta(\underline{u}_0) \ge \theta(\underline{u}_{-1})$, because then R, our final result, can be rewritten as

R: $p = (\underline{N}i: -1 \le i < 47: \theta(\underline{u}_{i+1}) < \theta(\underline{u}_i))$,

and Q_0 similarly as

Q_0: $p = (\underline{N}i: -1 \le i < j: \theta(\underline{u}_{i+1}) < \theta(\underline{u}_i))$ and $-1 \le j \le 47$.

We hardly have any choice, because $\underline{u}_{-1}$ must be defined such that $\theta(\underline{u}_{-1}) = 0$, ($\underline{u}_0$ could coincide with the negative y-axis), for instance $\underline{u}_{-1} = (0,-1)$.
Then the initialisation of a and α' results into:

```
ux,uy := 0,-1;  a := 0 .
```

To implement the extra points P_{47} and P_{48} we introduce an auxiliary variable h: $h = (j + 1) \underline{\text{mod}}\ 47$, by means of which we can round the arrays, so to speak.

Then the ultimate program becomes:

```
j,h,p := -1,0,0;  ux,uy := 0,-1;  a := 0;
do j ≠ 47 → j := j + 1;  h0,h := h,(j + 1)mod 47;
            vx,vy := x(h) - x(h0),y(h) - y(h0);  b := 0;
            do vx > 0  or  vy ≥ 0 → vx,vy := -vy,vx;  b := b + 1  od;
            if b < a  or  (b = a  and  vx*uy < ux*vy) → p := p + 1
             [] b > a  or  (b = a  and  vx*uy ≥ ux*vy) → skip
            fi;
            ux,uy := vx,vy;  a := b
od.
```

*

A final remark concerns the fact that during the analysis we switched our presentation from vectors (viz. $\underline{u}_i$) to angles (viz. $\theta(\underline{u}_i)$), and then back to vectors again. This may seem a needless detour, but: it has led to the discovery that in this particular problem vectors are conveniently represented as triples (viz. ux,uy,a), which is unusual and therefore not so trivial. This discovery was necessary to avoid a lot of case-analysis, which, during the examination,was hanging threateningly over our students' heads.

W.H.J. Feijen,
Eindhoven, January 6, 1978.