プログラミング及び実習 I ・第 6 回

2020 年 7 月 16 日

今回の内容

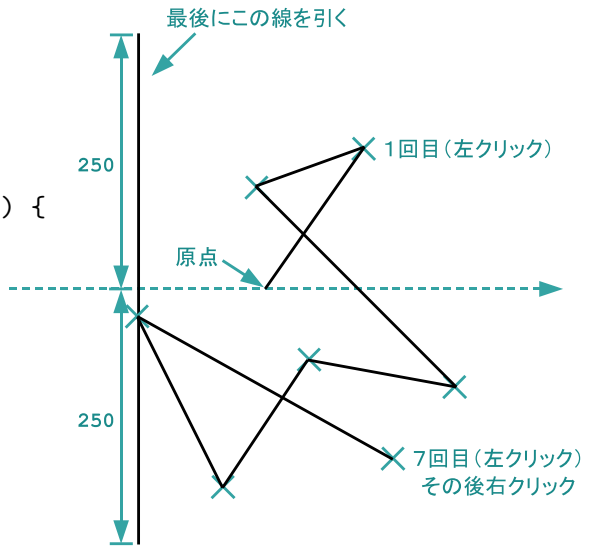
6.1 繰り返しの処理 (続き)

前回に続いて、もう少し複雑な繰り返しの処理と、それに付随するプログラミングの考え方を紹介します。

最小値や最大値の計算 繰り返しの処理の中で現れる数値の最小値や最大値を求めるプログラムを紹介します。次のプログラム minx.c は、ウィンドウ内をマウスで左クリックする度に、カメがクリックされた位置に向かって線分を描いていくプログラム[1]ですが、中ボタンや右ボタンが押されると、それまでにカメが動き回った点 (原点を含む) の中で最も左の点を通り、y 軸に平行な長さ 500 の線分を描きます。

minx.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int x, y, xmin;
 6
 7     xmin = 0;
 8     while (tGetClick() == 1) {
 9         x = tClickX();
10         y = tClickY();
11         tMoveTo(x, y);
12         if (x < xmin)
13             xmin = x;
14     }
15     tPenUp();
16     tMoveTo(xmin, -250);
17     tPenDown();
18     tMoveTo(xmin, 250);
19 }
```



このプログラムでは、変数 xmin に、それまでにカメがいた点の x 座標の**最小値**が保持されるようにしています。最初、カメは原点にいますから、変数 xmin は 0 で初期化しています (7 行目)。その後、while 文を使って、左クリックがされている限り

1. 左クリックされた点へ移動する
2. その点の x 座標が、変数 xmin に記憶されている値より小さければ、xmin の値を、その x 座標に更新する

[1] 各行の左端の数は、説明の都合で加えた行番号で、このプログラムの一部ではありません。

という 2 つの作業を繰り返しています。マウスの左ボタン以外がクリックされると while 文の実行は終了し、点 (xmin, −250) と点 (xmin, 250) が線分で結ばれます。

この minx.c というプログラムの

```
12              if (x < xmin)
13                  xmin = x;
```

の部分を

```
12              if (x > xmin)
13                  xmin = x;
```

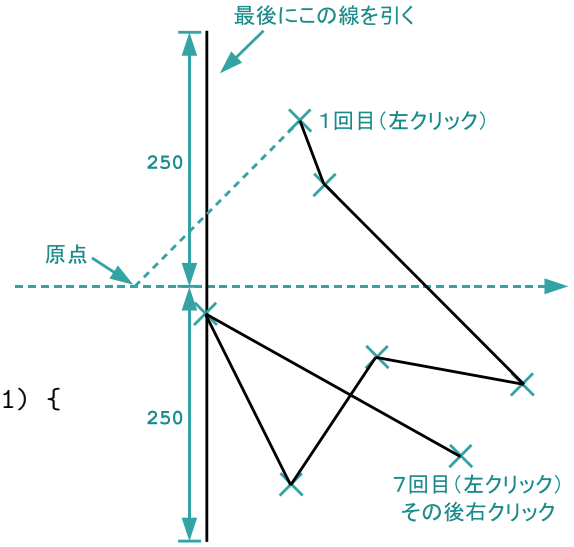
と変えれば、変数 xmin には、それまでにカメがいた点の x 座標の**最大値**が保持されるようになりますので、最も右の点を通る線分を引くことができます。ただし、この場合は、変数の名前は xmin ではなく、xmax 等にした方がよいのは言うまでもありません。

メモ

最初の値がわからない場合の最小値や最大値　minx.c では、カメの最初の位置である原点を含めて x 座標の最小値を求めていましたが、原点を含めず、左クリックされた点だけに関する x 座標の最小値を求めるためには、どのようなプログラムにしたらよいでしょうか。minx.c では、変数 xmin を 0 に初期化して、クリックされた点の x 座標に応じて minx の更新を繰り返せばよかったわけですが、クリックされた点のみを考える場合には、この方法は使えません。なぜなら、$x > 0$ の位置ばかりをクリックした時の最小値は正の値にならなければならないからです。この点に注意して、クリックされた点に限った x 座標の最小値を求めて線分を描くプログラムを考えると、たとえば、つぎのようなものになります。

minx2.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int x, y, xmin;
 6
 7     if (tGetClick() == 1) {
 8         x = tClickX();
 9         y = tClickY();
10         tPenUp();
11         tMoveTo(x, y);
12         tPenDown();
13         xmin = x;
14         while (tGetClick() == 1) {
15             x = tClickX();
16             y = tClickY();
17             tMoveTo(x, y);
18             if (x < xmin)
```



```
19                  xmin = x;
20          }
21          tPenUp();
22          tMoveTo(xmin, -250);
23          tPenDown();
24          tMoveTo(xmin, 250);
25      }
26 }
```

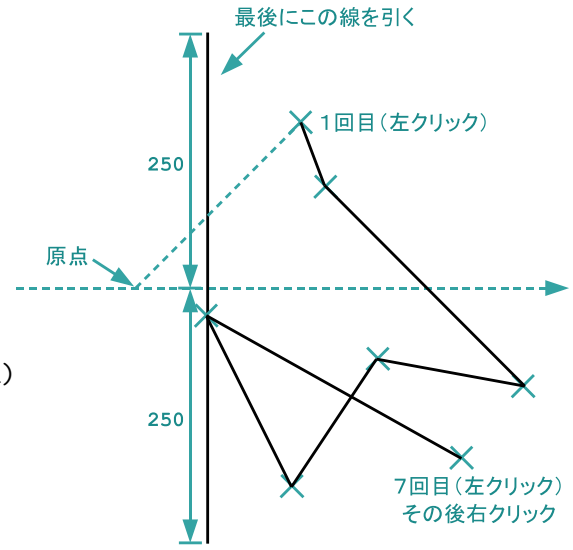
このプログラムでは、最初のクリックの処理を、while 文の外側で (繰り返しの処理の前に) 行い、その点の x 座標で minx を初期化しています。つまり、最初のクリックだけを特別扱いしているわけです。最初のクリックが左クリックでなかった場合は、1 つも左クリックされた点はなかったことになりますので、x 座標の最小値は決まりません。このような場合は、このプログラムは何もしない (線分は描かない) ようにしています。

メモ

この minx2.c と同じ事は、次のようなプログラムでも実現できます。minx3.c では、最初の左クリックを while 文による繰り返しの前に処理しておくのではなく、while 文の中で行うことにし、その代わり、それまでの左クリックの回数を変数 n で覚えておいて、n の値が 0 のとき (最初の左クリックの場合) を if 文を使って (14 行目と 15 行目) 特別扱いしています。

minx3.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int x, y, xmin, n;
 6
 7     tPenUp();
 8     n = 0;
 9     while (tGetClick() == 1) {
10         x = tClickX();
11         y = tClickY();
12         tMoveTo(x, y);
13         tPenDown();
14         if (n == 0 || x < xmin)
15             xmin = x;
16         n = n + 1;
17     }
18     if (n > 0) {
19         tPenUp();
20         tMoveTo(xmin, -250);
21         tPenDown();
22         tMoveTo(xmin, 250);
23     }
24 }
```



最初の左クリックでは、n の値は 0 ですから、クリックされた点の x 座標や xmin の値が何であっ

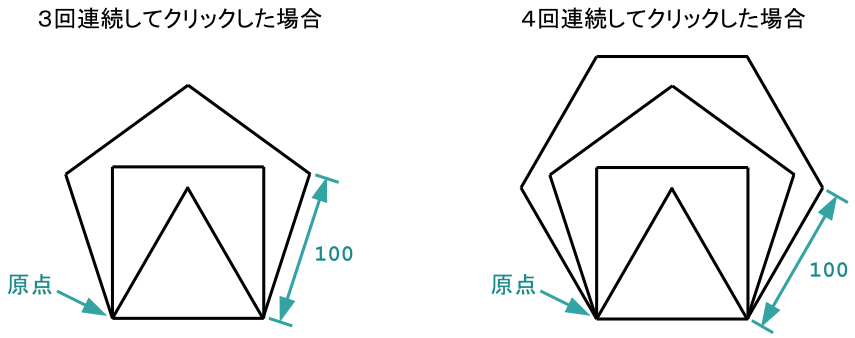
ても `n == 0 || x < xmin` という条件は成り立ち、変数 `xmin` が、その x 座標で初期化されます。2 回目以降の左クリックでは、`n == 0` は成り立ちませんので `x < xmin` の場合のみ `minx` が上書き (最小値が更新) されます。

最初に現れる値がわからないときに最大値を求めたい場合も、`minx2.c` や `minx3.c` と同様な考え方で実現することができます。

メモ

二重の繰り返し　つぎのようなプログラムを書くことを考えます。

- マウスを n 回連続してクリックすると、下図のように 1 辺が 100 の正三角形、正方形、正五角形、...、正 $(n+2)$ 角形を重ねて描くプログラム



このプログラムが行わなければならない仕事は、おおよそ次のようなものになります。

1. マウスボタンがクリックされるのを待ち、クリックの連続した回数を取得する。
2. その回数を n とすると、$k=3,\ 4,\ \ldots,\ n+2$ の各 k について、1 辺が 100 の正 k 角形を描く。

この内、1 は `tGetClick` と `tClickCount` を使い、適当な変数 (たとえば) `n` にクリックの回数を格納することで実現できます。一方、2 については、`while` 文を使った繰り返しの処理で実現できそうです。このプログラムの概形は、次のようなものになるでしょう。

```
int k, n;

tGetClick();
n = tClickCount();
k = 3;
while (k <= n + 2) {
    1辺が100の正 k 角形を描く
    k = k + 1;
}
```

さらに、この while 文の中の「1 辺が 100 の正 k 角形を描く」という仕事の内容をもっと詳しく書くと、tForward(100); と tTurn(360.0 / k); を k 回繰り返し実行するということになりますので、この部分もまた while 文を使った繰り返しの処理になります。つまり、繰り返しの処理で繰り返される作業もまた繰り返しの処理を含んでいるわけです。この内側の繰り返しの部分は、整数値の変数 (たとえば) i を用意 (宣言) しておいて

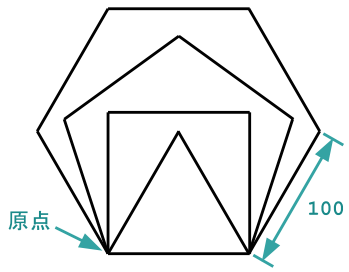
```
i = 0;
while (i < k) {
    tForward(100);
    tTurn(360.0 / k);
    i = i + 1;
}
```

のようなプログラムで実現できます。以上のような考え方に沿ってプログラム全体を書くと、つぎのようなものができあがります。

nested.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int i, k, n;
 6
 7     tGetClick();
 8     n = tClickCount();
 9     k = 3;
10     while (k <= n + 2) {
11         i = 0;
12         while (i < k) {
13             tForward(100);
14             tTurn(360.0 / k);
15             i = i + 1;
16         }
17         k = k + 1;
18     }
19 }
```

4回連続してクリックした場合

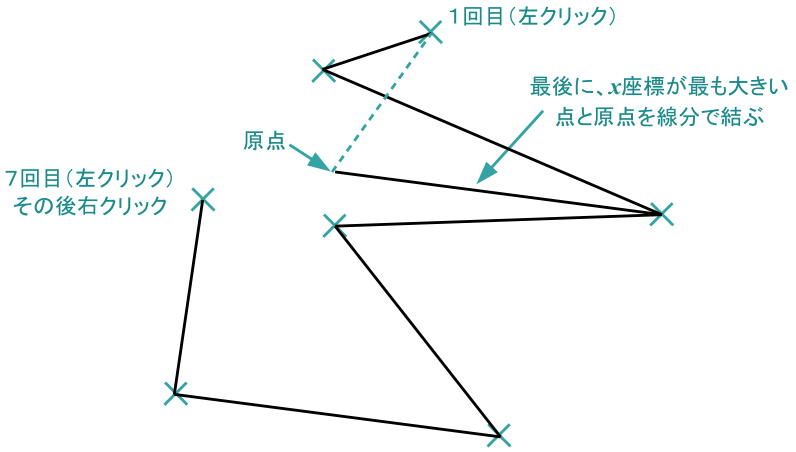


このプログラムで、変数 k は「今、何角形を描こうとしているのか」を、変数 i は「今 (その正多角形の) 何辺目までを描き終えたか」を、それぞれ覚えていますので、この 2 つの変数を 1 つで済ますわけにはいきません。また、11 行目の i = 0; は、それぞれの正多角形を描き始める前に毎回実行しなければなりませんので、これを外側の while 文の前 (たとえば、9 行目と 10 行目の間) に移動することもできないことに注意してください。

メモ

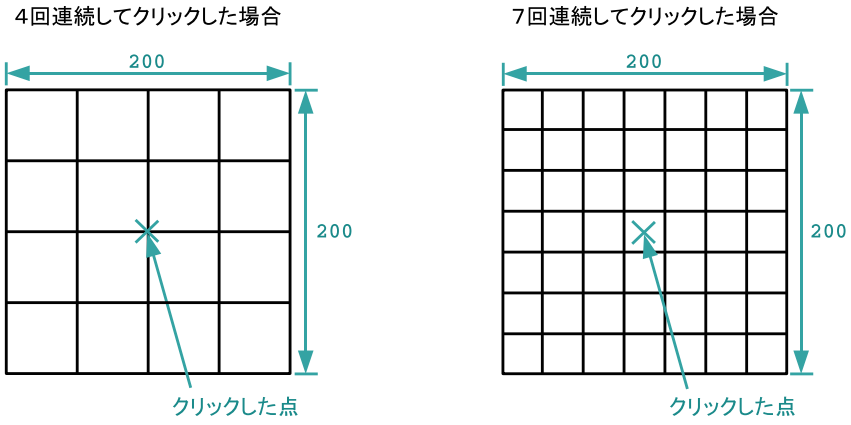
6.2 演習問題

1. 左クリックした点をカメが次々と線分で結んで行き、中ボタンや右ボタンを押すと、それまでに左クリックした点の内、x 座標が最大の点と原点を線分で結ぶようなプログラム `rightmost.c` を作り、コンパイル、実行して、正しく動作することを確認しなさい。最初のクリックが中ボタンや右ボタンであった場合は、このプログラムは何の仕事もしません。左クリックされた点の中で x 座標が最大の点が複数ある場合は、それらの中で y 座標が最大の点を選んで、その点と原点を線分で結ぶようにしてください。



ヒント: `minx3.c` を参考にしましょう。左クリックされた点の内、原点と結ばれることになる点の候補の座標を常に記憶するようにしておき、最後に、原点とその座標を線分で結びましょう。最初に左クリックされた点は、まずこの候補となります。ある時点で候補となっている点より x 座標が大きい点が左クリックされるとその点が新しい候補となります。x 座標が同じ場合でも、y 座標がより大きければ、やはり新しい候補となります。

2. ウィンドウ内を n 回連続してマウスクリックすると、クリックされた位置を中心に、幅 200、高さ 200 の正方形を $n \times n$ に分割し、図のような格子を描く C プログラム `mesh.c` を作り、コンパイル、実行して、正しく動作することを確認しなさい。1 回だけクリックされた場合は、幅 200、高さ 200 の正方形を描くことになります。



ヒント: この図形の描き方に迷ったら、たとえば次のような手順で描きましょう。

(1) ペンを上げて、描きたい図形全体の左下端に tMoveTo 関数を使って移動する。

(2) while 文を使って、以下の手順を n 回繰り返す。

(2-1) 右 (x 軸の正の向き) を向く。特定の向きにカメの向きを変えるには tTurnTo という関数を使うことができます。tTurnTo(t); を実行することで、カメは x 軸の正の向きを基準として反時計回りに t° の方向へ向きを変えます。つまり、tTurnTo(0); を実行すれば、カメは x 軸の正の向きになります。

(2-2) while 文を使って、(2-2-2) までの手順を n 回繰り返す。

(2-2-1) 一辺が $\frac{200}{n}$ の正方形を描く。

(2-2-2) 描いた正方形の右下の頂点に移動して、(そのとき右を向いていなければ) 右を向く

(2-3) tBackward 関数で 200 だけ後戻りする。

(2-4) 上 (y 軸の正の向き) を向く。

(2-5) tForward 関数で $\frac{200}{n}$ だけ進む。

この手順では、2 重の繰り返し処理を行っていることに注意しましょう (正方形の 4 辺を描く際にも繰り返しの処理を使う場合は、3 重の繰り返しになります)。

6.3 今回の実習内容

1. プリントをもう一度読み返しましょう。minx3.c と nested.c の 2 つの例題については、ソースプログラムをそれぞれ作成し、コンパイル、実行して、正しく動作することを確認してください。プログラムが完成したら「課題の提出と確認」の Web ページから提出してください。
2. 演習問題に取り組みましょう。それぞれのプログラムが完成したら、「課題の提出と確認」の Web ページからの提出を忘れないでください。
3. クイズに答えてください。前回までと同様に「課題の提出と確認」の Web ページで「第 6 回クイズ」を選択し、「送信」のボタンをクリックしてクイズに答えてください。

プログラミング及び実習 I ・第 6 回・終わり

付録: 前回の演習問題のプログラム例

dpolygon2.c

```
#include <turtle.h>

main ()
{
    int n, i;

    tGetClick();
    n = tClickCount();
    i = 0;
    while (i < n * 10) {
        if (i % 2 == 0) {
            tPenDown();
            tForward(13);
        }
        else {
            tPenUp();
            tForward(7);
        }
        i = i + 1;
        if (i % 10 == 0)
            tTurn(360.0 / n);
    }
}
```

margin.c

```
#include <turtle.h>

main ()
{
    int x, y;

    while (tGetClick() == 1) {
        x = tClickX();
        y = tClickY();
        if (-100 <= x && x = 200 && -200 <= y && y <= 100)
            tMoveTo(x, y);
    }
    tPenUp();
    tMoveTo(0, 0);
}
```

farther.c

```
#include <turtle.h>

main ()
{
    int x, y, dd;

    dd = 0;
    while (tGetClick() == 1) {
        x = tClickX();
        y = tClickY();
        if (dd <= x * x + y * y) {
            tMoveTo(x, y);
            dd = x * x + y * y;
        }
    }
    tPenUp();
    tMoveTo(0, 0);
```

```
}
```