プログラミング及び実習 I ・第 2 回

2024 年 6 月 13 日

今回の内容

2.1 C プログラムの文

これまで見てきた C プログラムは、プログラム中に書かれている指示をそこに書かれている順に実行させていくものでした。この作業の 1 つ 1 つは、`tForward` を使ったカメの移動であったり、`=` を使った変数への値の代入であったりしましたが、C プログラムでは、これらのようにひとまとまりの作業を行わせるための指示を**文**[1]と呼びます。たとえば、前回の例題 `var1.c` の中に現れているつぎの 6 行は、それぞれ C プログラムの文になっています。

```
a = 300 / 1.41421;
tForward(a);
tTurn(135);
tForward(300);
tTurn(135);
tForward(a);
```

C プログラムのもっとも基本的な文はすべて「`;`」(セミコロン) で終ります。上の例では 1 つの行に 1 つの文を書いてありますが、複数の行にわたって 1 つの文を書いたり、また複数の文を 1 つの行にまとめて書いてもかまいません。どのように書いても C プログラムとしては同じ意味になります。たとえば上の 6 行をつぎのように書いてもまったく同じ意味です。

```
          a
        = 300
       / 1.41421
     ;tForward (a
   );  tTurn  (135)
  ;  tForward ( 300 );
tTurn(135);tForward (a);
```

ただし、`1.41421` や `tForward`、`tTurn` などは、この一連の文字 (や数字、記号) の列で、ひとまとまりの意味を持ちますので、これを複数の行に分割して書くことはできません。また、プログラムとしてたとえ同じ意味であっても、後でプログラムを読んで理解したり、保守・修正したりするためには、人間にとって**読みやすく書いておく**ことが重要です。プログラムは単に動けばよいというものではないことに注意しましょう。

メモ

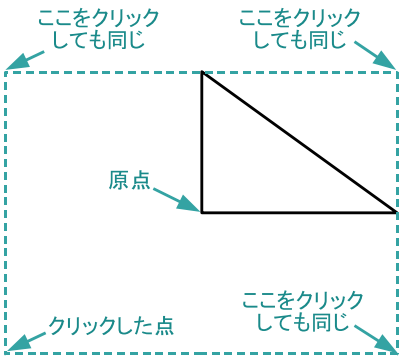
[1] 英単語 statement の訳語です。

2.2　場合分けの処理

決まりきった作業を常に同じように行うのではなく、時と場合によって違った作業をプログラムに行わせたいことがあります。C ではこのために **if 文**とよばれる形の文を使うことができます。つぎのプログラムは、マウスでクリックされた座標が $(x,\ y)$ であったときに、原点 $(0,\ 0)$ と $(|x|,\ 0)$、$(0,\ |y|)$ の 3 点を頂点とする直角三角形を描きます。

abs1.c

```
#include <turtle.h>

main()
{
    int x, y;

    tGetClick();
    x = tClickX();
    y = tClickY();
    if (x < 0)
        x = -x;
    if (y < 0)
        y = -y;
    tMoveTo(x, 0);
    tMoveTo(0, y);
    tMoveTo(0, 0);
}
```



まず、注目して欲しいのは

```
    if (x < 0)
        x = -x;
```

の部分です。ここでは、変数 x の値が負の時 (0 より小さい時) に限って `x = -x;` を実行しています。変数 x の値が負でない (0 以上の) 時には、`x = -x;` は実行されません。続く、

```
    if (y < 0)
        y = -y;
```

では、変数 y について同じ作業をしています。

メモ

if 文の書き方　ここで使われている `if` は C の予約語 (特別な意味を持った単語[2]) で、このように特定の条件が成立した場合にのみ、特定の作業を行わせることができます。この形の文を **if 文**と呼びます。if 文の一般的な形は

```
if ( 条件式 )
    文
```

[2]前回説明したように、予約語である `if` を変数の名前として用いることはできません。

で、「条件式」が成り立っている時に限って「文」の部分が実行されます。「条件式」が成り立っている時にのみ実行される「文」の部分を、この `if` 文の **then 節**と呼びます。C プログラムでは、複数の行にわたって文を書いても 1 行にまとめて文を書いても同じ意味になりますから、この `if` 文全体を

```
if ( 条件式 ) 文
```

のように 1 行に書いても同じ意味になります。また、条件式を囲っている (や) の両側には空白があってもなくてもどちらでもかまいません。

メモ

if 文の条件式　`if` 文ではいろいろな条件を指定することができますが、ここではごく基本的なものだけを紹介します。a や b の部分には任意の数式を書くことができます。それぞれの数式がまず計算されてその結果が比較されます。

条件式	条件式の意味
a == b	a と b の値が等しい
a < b	a の値が b より小さい
a <= b	a の値が b 以下である

条件式	条件式の意味
a != b	a と b の値が等しくない
a > b	a の値が b より大きい
a >= b	a の値が b 以上である

これらの条件式は a や b が実数値でも整数値でも使うことができます。一方が実数値でもう一方が整数値の場合は、四則演算の場合と同じように、整数値が実数値に変換されてから比較が行われます。コンピュータの中の実数値はあくまで近似値でしかありませんから、多くの場合、プログラム中に a == b や a != b と書いて 2 つの実数値が等しいかどうかを調べることには意味がありませんので注意が必要です。たとえば、実数値の変数 x、y があったとき、`x/y*y == x` が常に成り立つとは限りません。

残念ながら C 言語では、これらの ==、!=、<、<=、>、>= は 2 つの値を比較することしかできませんので「a < b < c」や「a < b, c」、「a > 0, b < 0」のような書き方はできません。このように書くと、別の意味になってしまいますので注意してください。「a < b と b < c が共に成り立つ」といったように複数の条件を組み合わせる方法については第 4 回に勉強する予定です。

メモ

重なった if 文　if 文はその一部として (条件が成り立っている時にのみ実行される) 文 (then 節) を持っていますが、if 文全体としても 1 つの文となります。ですから

```
if (条件式1)
    if (条件式2)
        文
```

のように「条件式$_1$」が成り立っている時に限って実行する文 (then 節) が、また、別の if 文となることもできます。この例の場合、結局「条件式$_1$」と「条件式$_2$」が共に成り立っている時のみ「文」が実行されます。どちらかの条件が成り立っていない場合には「文」は実行されません。

メモ

複数の文を条件付きで実行する　条件付きで実行したい文が複数ある場合には、それらの文を { と } で囲って

```
if ( 条件式 ) {
    文1
    文2
     ⋮
    文n
}
```

のように書きます。「条件式」が成り立っている時のみ「文$_1$」...「文$_n$」が順に実行されます。「条件式」が成り立っていない場合はこれらの文は実行されません。このようにいくつかの文を { と } で囲ってひとまとまりにしたものを**ブロック**と呼びます。ブロックはあたかもそれが 1 つの文であるかのように扱われます。この例では、どこからどこまでの文が条件付きで実行されるのかが見やすいように「if (条件式)」がかかっている部分を**字下げ**して (行の先頭から間を空けて) 書いています[3]。どのように書いてもプログラムとしては同じ意味になりますが、このように書いておくことで、自分の書いたプログラムがより理解しやすいものになり、プログラムの誤りも少くなります。ただし、いくら字下げして書いても、{ や } を書かずに

```
if ( 条件式 )
    文1
    文2
     ⋮
    文n
```

[3]多くのエディタでは、TAB キーを押すと、4 文字あるいは 8 文字分の字下げがなされますので、この機能を使って字下げすることをおすすめします。

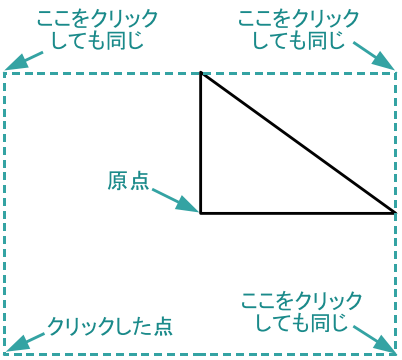
と書いてしまうと、条件付きで実行される then 節は「$文_1$」だけになってしまい、「$文_2$」...「$文_n$」は「条件式」が成り立っていない場合にも実行されてしまいますので注意が必要です。「条件式」がかかるのは「if (条件式)」の直後の文一つだけであり、それより後の文は、この if 文とは独立な文と見なされます。

メモ

else 付きの if 文　条件が成り立っている時と成り立っていない時とで違った作業をそれぞれ行うこともできます。つぎの abs2.c というプログラムは、abs1.c と同じことを行うプログラムですが、これを別のやり方で実現しています。

abs2.c

```
#include <turtle.h>

main()
{
    int x, y;

    tGetClick();
    x = tClickX();
    y = tClickY();
    if (x < 0)
        tMoveTo(-x, 0);
    else
        tMoveTo(x, 0);
    if (y < 0)
        tMoveTo(0, -y);
    else
        tMoveTo(0, y);
    tMoveTo(0, 0);
}
```



この abs2.c をコンパイルして実行してみると、abs1.c の時とまったく同じように動きます。このプログラムで使われている else も C の予約語の 1 つで、if と共に用いられます。else を伴う if 文の一般的な形は

```
if ( 条件式 )
    文1
else
    文2
```

です。もちろん、この 4 行を 1 行で (あるいは 2 行や 3 行で) 書てもかまいません。「条件式」が成り立っている場合には「$文_1$」が実行され、成り立っていない場合には「$文_2$」が実行されます。「$文_1$」や「$文_2$」はブロックであったり、あるいはそれ自体が if 文であったりしてももちろんかまいません。「条件式」が成り立っている時にのみ実行される「$文_1$」を (else がない場合とおなじように) こ

の if 文の **then 節**と呼び、「条件式」が成り立っていない時にのみ実行される「文$_2$」を、この if 文の **else 節**と呼びます。

else は、{ と } を使って明示的にその対応が分かるようプログラムが書かれていない場合はもっとも近い if を対をなしますので、たとえば

if（条件式$_1$） if（条件式$_2$）文$_1$ else 文$_2$

のようにプログラムを書くと

if（条件式$_1$）{ if（条件式$_2$）文$_1$ else 文$_2$ }

と書いたのと同じ意味になります。この else を最初の if と対にしたい場合は

if（条件式$_1$） { if（条件式$_2$）文$_1$ } else 文$_2$

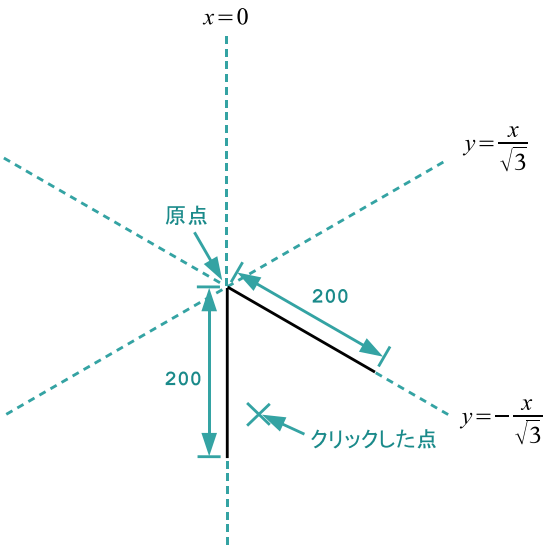
のように書く必要があります。

メモ

3つ以上の場合分け else 付きの if 文の then 節や else 節を、さらに if 文とすることで、3 つ以上の場合の場合分けを実現することができます。つぎは、マウスでクリックした点が、平面を 3 つの直線 $x = 0$、$y = \dfrac{x}{\sqrt{3}}$、$y = -\dfrac{x}{\sqrt{3}}$ で分割してできる 6 つ領域のどこに属するかを、原点を端点とした 2 つの線分を描いて示すプログラムです。

slice.c

```
#include <turtle.h>

main()
{
    int x, y, t;

    tGetClick();
    x = tClickX();
    y = tClickY();

    if (x >= 0) {
        if (y > x / 1.73205)
            t = 30;
        else if (y > - x / 1.73205)
            t = -30;
        else
            t = -90;
    }
    else {
        if (y < x / 1.73205)
            t = -150;
```



```
        else if (y < - x / 1.73205)
            t = 150;
        else
            t = 90;
    }

    tTurn(t);
    tForward(200);
    tTurn(120);
    tPenUp();
    tForward(200);
    tTurn(120);
    tPenDown();
    tForward(200);
}
```

このプログラムでは、まず クリックされた点の x 座標が 0 以上であるか負であるかで 2 通りに場合分けし、それぞれの場合を、さらに、直線 $y = \dfrac{x}{\sqrt{3}}$ と直線 $y = -\dfrac{x}{\sqrt{3}}$ で分けられた領域のどこにあるかで 3 通りに場合分けしています。

```
if (y > x / 1.73205)
    t = 30;
else if (y > - x / 1.73205)
    t = -30;
else
    t = -90;
```

の部分では、$x \geqq 0$ の場合に、(x, y) が 3 つの領域のどれに属するかを判定しています。C というプログラミング言語に、あたかも

```
else if ( 条件式 ) 文
```

という形の構文があって、それを使っているようにも見えますが、実はそうではありません。場合分けに関するプログラマの意図が理解しやすいように、プログラムの書式を工夫しているだけです。このプログラムを、if 文の構造がよく分かるように then 節や else 節を { } で括って書くと、

```
if (y > x / 1.73205) {
    t = 30;
}
else {
    if (y > - x / 1.73205)
        t = -30;
    else
        t = -90;
}
```

のようになります。else if と続くのは、単に、else 付きの if 文 の else 節 (条件が成り立っていないとき実行される文) が、また (else 付きの) if 文 の形をしているからに過ぎません。

変数の代入の性質を考えると、この部分は

```
t = -90;
if (y > x / 1.73205)
    t = 30;
else if (y > - x / 1.73205)
```

```
        t = -30;
```

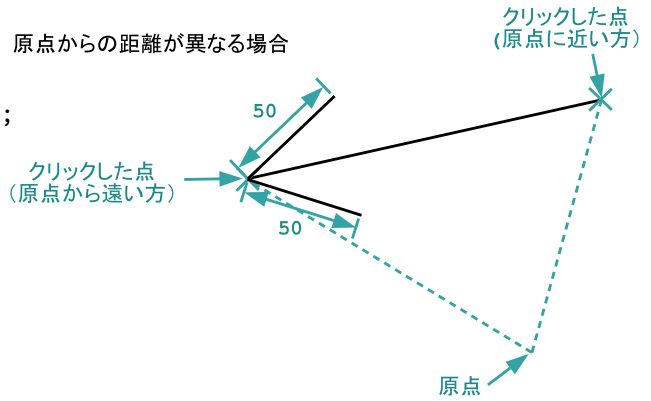
と書いても同じ動作となりますが、この場合は、else 付きの if 文 の else 節が else なしの if 文 の形になっています。

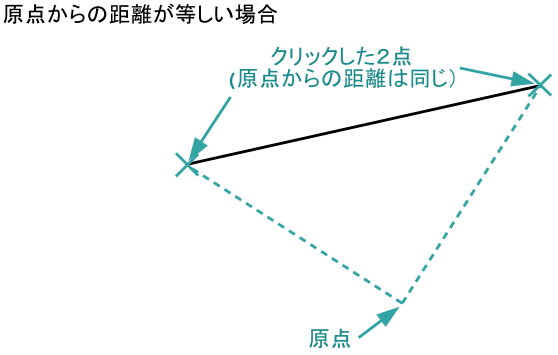
メモ

if 文の応用例　ここで、if 文を使った場合分けの処理を応用したプログラムの例を見てみましょう。次のプログラム outward.c は、2 点をクリックすると、原点に近い方の点から遠い方の点へ矢印を描きます。ただし、2 つの点が原点から同じ距離にある場合は、矢印ではなく線分で結びます。

outward.c

```
#include <turtle.h>

main()
{
    int x1, y1, x2, y2, dd1, dd2, t;

    tGetClick();
    x1 = tClickX();
    y1 = tClickY();
    dd1 = x1 * x1 + y1 * y1;

    tGetClick();
    x2 = tClickX();
    y2 = tClickY();
    dd2 = x2 * x2 + y2 * y2;

    if (dd1 > dd2) {
        t = x1;
        x1 = x2;
        x2 = t;
        t = y1;
        y1 = y2;
        y2 = t;
    }

    tPenUp();
    tMoveTo(x1, y1);
    tPenDown();
    tMoveTo(x2, y2);

    if (dd1 != dd2) {
        tTurn(150);
        tForward(50);
        tTurn(120);
        tPenUp();
        tForward(50);
        tTurn(120);
        tPenDown();
```





```
            tForward(50);
        }
    }
```

このプログラムでは、まず、クリックされた 2 点の座標を取得して、それぞれ、変数 x1、y1 の組と x2、y2 の組で記憶し、原点からの距離の 2 乗を計算して、変数 dd1 と dd2 にそれぞれ代入しています。続く if 文では dd1 > dd2 という条件を調べ、この条件が成り立っているときだけ、

```
t = x1;
x1 = x2;
x2 = t;
t = y1;
y1 = y2;
y2 = t;
```

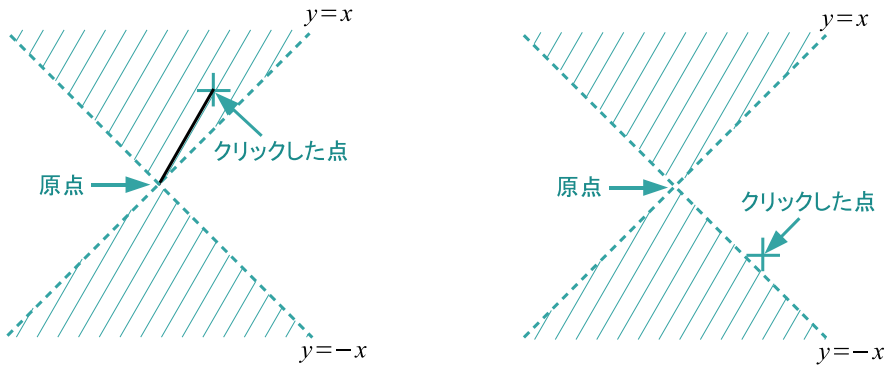
を実行して、変数 x1、y1 の組と x2、y2 の組で記憶されている 2 つの点の座標を交換しています。変数 x1 と x2 の内容を交換するためには、まず x1 の値を、別に用意しておいた変数 t にコピーしておき、x1 には x2 の値を格納します。続いて x2 に (変数 t に記憶しておいた) 変数 x1 の古い値を格納すれば交換完了です。同じことを、変数 y1 と y2 について行えば、座標の交換が完了します。この if 文が実行されることで、変数 x1 と y1 には、原点に近い方の点の座標が、x2 と y2 には、原点から遠い方の点の座標が格納された状態になります。もし、初めからそうであった場合や、距離が等しい場合は、この if 文は何もしません。

このプログラムは、続いて、点 (x1, y1) から点 (x2, y2) に向けて線分を描き、さらに、dd1 と dd2 の値が異なる場合のみ矢印の先端を描きます。

メモ

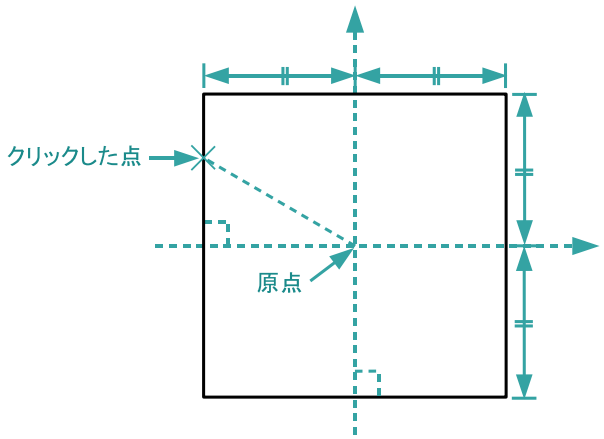
2.3 演習問題

1. 原点以外の 1 点をクリックすると、クリックした点の座標が、$|y| \geqq |x|$ を満たしている場合だけ、原点からその点までの線分を描く C プログラム `diagonal.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。このプログラムでは、クリックされた点の x 座標と y 座標を比較しなければいけませんが、この時 (誤差の影響を受けないように) 整数値と整数値の比較になるように工夫してください。



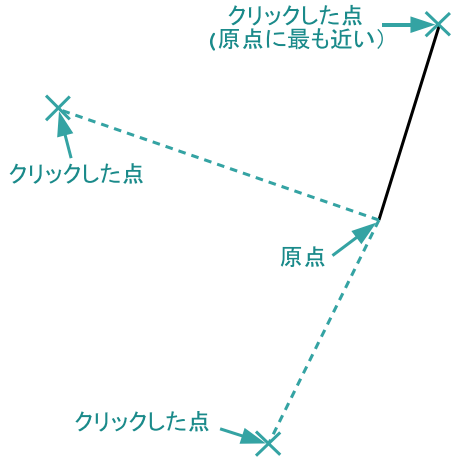
注意: このプログラムが正しく動作するかどうかをテストするためには、マウスでクリックしようとしている座標を正確に知る必要があります。プログラムを起動して、コントロール (Ctrl) キーを押した状態にすると、ウィンドウの左上にマウスポインタの座標が表示されるとともに、マウスの動きが遅くなって、特定の座標の点をクリックしやすくなります。この機能を使ってプログラムをテストしてください。

2. 原点以外の 1 点をクリックすると、その点を辺上に含む、図のような正方形を描く C プログラム `box.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。



ヒント: クリックされた点の座標を (x, y) とすると、この正方形の 1 辺の長さは、$|x|$ と $|y|$ の最大値の 2 倍であることに注意しましょう。

3. 原点以外の 3 点をクリックすると、その内、原点に最も近い点に向けて、原点から線分を描く C プログラム `nearest.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。原点に最も近い点が複数ある場合は、そのいずれかの 1 点 (どれでも ok) と結ぶようにしてください。



ヒント: 本来なら、2 点間の距離を求めるには平方根の計算が必要となりますが、このプログラムでは、平方根の計算は必要ありません。距離の大小を比較するためには、距離の 2 乗の値の大小を比較すればよいことに注意してください。

2.4 今回の実習内容

1. このプリントをゆっくり読み返しましょう。`slice.c` と `outward.c` の 2 つの例題プログラムについては、それぞれソースプログラムを作成し、コンパイル、実行してみて正しく動くかどうか確認しましょう。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. 10 ページの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、やはり「課題の提出と確認」の Web ページから提出してください。プログラムは適当な字下げをして書くようにしてください。
3. クイズに答えてください。前回と同様に「課題の提出と確認」の Web ページで「第 2 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 2 回・終わり