講習会

Arduino

5回目

制御分岐

目的

- 今まで紹介した関数でもプログラムを組むことはできるが、あくまで上から順番に読むことしかできない。
- 処理分岐の関数を入れることでどういう条件で何をしたいのか等のプログラムを組むことができる。
- プログラムの幅が広がるため積極的に活用していこう。
- マイコンカーはほぼif文とswitch文を使っている。

- ここでは処理分岐に関する関数と例を紹介する。

処理分岐の関数

- If
- If else
- switch case
- for
- while
- do while

if

- ifは、式と一緒に用いられ、ある条件、例えば入力が特定の値であるか、が満たされているかを確認する。

使用例

- if (somveVariable > 50){
- 　// ここで何か行う
- }

- このプログラムは、someVariableが50よりも大きいかどうかを比較する。もし条件が満たされていれば、プログラムはあるコードを実行する。言い換える と、丸括弧内の式が真の場合、波括弧内の命令が実行される。もしそうでなければ、その命令は実行されない。

- if 文の後の波括弧は省略されることがある。この場合は、その次の行(セミコロンによって区切られる)だけが条件によって実行される命令となる。

- if (x > 120) digitalWrite(LEDpin, HIGH);
-
- if (x > 120)
- digitalWrite(LEDpin, HIGH);
-
- if (x > 120){ digitalWrite(LEDpin, HIGH); }
-
- if (x > 120){
- digitalWrite(LEDpin1, HIGH);
- digitalWrite(LEDpin2, HIGH);
- }

- 丸括弧の中で評価される式では、1個以上の関係演算子や比較演算子を利用することが多い。

等価演算子と関係演算子

- x == y (xとyが等しい)
- x != y (xとyは等しくない)
- x < y (xはyより小さい)
- x > y (xはyより大きい)
- x <= y (xはy以下)
- x >= y (xはy以上)

注意

- 例えば、if (x = 10)のように、「=」を誤って使わないように注意すること。「=」は、単純代入演算子であり、前の例ではxを10に設定する。この場合は、if(x== 10)のように、等価演算子である「==」を使い、xが10かどうかを調べる必要がある。後者(x == 10)は、xが10のときだけ真であるが、前者(x = 10)は、常に真である。
- これは、C言語がif(x = 10)という文を以下のように評価することによる。最初に、xに10を代入する(=は単純代入演算子である)。このため、今xは10である。　if文が10を評価する。そして、ゼロ以外の数字は真と評価されるため、これは常に真である。よってif(x = 10)は、常に真であり、if文を使うときの結果としては好ましくないものである。さらに、xには10が代入され、これも好ましくない結果である。
- if文はif...else構造での分岐の一部としても用いられる。

if else

- if...elseは、複数の比較をグループ化することにより、基本的なifよりもより広くコードの流れを制御する。
- elseの後には別のifを記述することができる。これにより、複数の相互排他の比較を順に実行することができる。
- 比較結果が真になるまで順に比較を行っていく。真になる比較が見つかると、その比較に関連したブロックのコードが実行され、それ以降のif/elseの実行されない。どの比較も真にならない場合、elseブロックがあれば実行される。
- else ifブロックは、elseブロックがあってもなくても利用することができる。else ifブロックの数に制限はない。
- 分岐や複数の相互排他の比較を評価するには、switch caseを利用することもできる。

使用例

- 例えば、アナログ入力を比較し、入力値が500よりも小さければあることを行い、500以上であれば別のことを行うということができる。コードは以下のようになる。if による条件確認のための書式は以下の通り。

- if (pinFiveInput < 500){
- 　// action A
- }
- else{
- 　// action B
- }

else if を使った例

- if (pinFiveInput < 500){
- 　// do Thing A
- }
- else if (pinFiveInput >= 1000){
- 　// do Thing B
- }
- else{
- 　// do Thing C
- }

switch case

- if文と同じように、さまざまな状況に応じて異なるコードを実行することで、switch文はプログラムの流れを制御する。switch文は、制御式とcaseラベルとを比較する。制御式の値とcaseラベルの式とが等しいとき、そのcaseラベルに付随するコードを実行する。
- 制御式の値と等しいcaseラベルがない場合、defaultラベルがあれば、そのラベル付き文に制御が移る。
- break文はswitch文の実行を終了する。そのため、通常は、caseラベルに付随するコードの最後に表れる。caseラベルに付随するコードの終わりにbreak文がない場合、switch文は、次のbreakが見つかるまで、もしくは、switch文の最後に到達するまでコードを実行する(これをfall-through(通り抜け)と呼ぶ)。

書式

- switch(式) {
- 　case ラベル1:
- 　　// 文
- 　　break;
- 　case ラベル2:
- 　　// 文
- 　　break;
- 　default:
- 　　// 文
- }

使用例

- switch (var) {
- 　case 1:
- 　　// varが1のときに実行するコード
- 　　break;
- 　case 2:
- 　　// varが2のときに実行するコード
- 　　break;
- 　default:
- 　　// どのcaseラベルにも一致しないときに実行するコード
- 　　// defaultは、省略可能
- 　}

for

- for文は、波括弧で囲まれた文のブロックの繰り返しに使われる。増分カウンタがループの実行制御に通常利用される。for文は操作を繰り返すときに有用であり、配列と組み合わせてデータやピンの集合に対する操作を行うときに利用することが多い。
- for文は3つの部分に分けられる。
- for(初期化(節1); 制御式(式2); 増分操作(式3)) {
- 　//statement(s);
- }
- 初期化(節1)は最初に一度だけ実行される。ループの最初に制御式(式2)が評価され、その値が真であった場合、ループ本体が実行される。その後、増分操作(式3)が実行され、制御式(式2)が評価される。制御式が偽になったとき、ルー プは終了する。
- for (節1; 式2; 式3)　文

```
// PWMピンを使って、LEDをほのかに点灯させる
int PWMpin = 10; // 10番ピンに、470Ωの抵抗とLEDとを直列につなぐ。

void setup(){
  // no setup needed
}

void loop(){
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

ヒント

- C言語のforループは、BASICのような他の言語に見られるforループよりも柔軟である。for文のどれかもしくはすべての部分は省略可能である(セミコロンは必要)。また、初期化や制御式、増分操作にはC言語で有効などのような式でもよく、また、浮動小数を含むどんな型や、直接は関係のない変数を使ってもいい。このfor文の独特の方式により、まれに表れるプログラミング上の問題を解決できることもある。

- 例えば、増分操作として、掛け算を使うと、等比数列を生成することができる。以下のプログラムは、2,3,4,6,9,13,19,28,42,63,94を表示する。

- for(int x = 2; x < 100; x = x * 1.5){
- println(x);
- }

一つのfor文で、LEDを明るくしたり暗くする例

```
void loop(){
   int x = 1;
   for (int i = 0; i > -1; i = i + x){
      analogWrite(PWMpin, i);
      if (i == 255) x = -1;             // switch direction at peak
      delay(10);
   }
}
```

while

- while文では、制御式が偽になるまでループ本体を実行し続ける。制御式で評価される値を変更しなければ無限に実行される。これは、変数を増加させたり、センサーの値を評価したりすることによって実現される。
- 評価式はループ本体を実行する前に評価される。このため、ループ本体を1回も実行しないことがある。

使用例

- var = 0;
- while(var < 200){
- 　// 200回繰り返す
- 　var++;
- }

do while

- do文は、制御式がループの最後に評価されることを除き、while文と同様の動作をする。このため、doループは最低でも1回は実行される。

使用例

```
do
{
  delay(50);          // センサーが安定するまで待つ
  x = readSensors();  // センサーを調べる

} while (x < 100);
```

処理分岐時の動作に関する関数

- break
- continue
- return

break

- break文は、do文やfor文、while文の実行を、制御式による評価を行わずに、終了させる。switch文の実行も終了させる。

使用例

```
for (x = 0; x < 255; x ++){
    digitalWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > threshold){      // bail out on sensor detect
       x = 0;
       break;
    }
    delay(50);
}
```

continue

- continue文は、現在のdo文やfor文、while文による繰り返し実行のループの残りを実行せず、ループの制御式の評価へと制御を移し、次の繰り返し実行のループを実行する。

使用例

```
for (x = 0; x < 255; x ++){
    if (x > 40 && x < 120){      // create jump in values
        continue;
    }
    digitalWrite(PWMpin, x);
    delay(50);
}
```

return

- 関数を終了し、必要ならば、呼び元の関数に値を返す。

使用例

- センサーからの入力と閾値とを比較する関数。

- int checkSensor(){
- 　if (analogRead(0) > 400) {
- 　　return 1;
- 　else{
- 　　return 0;
- 　}
- }

- バグを含む多くのコードをコメントアウトせずにテストするときにも簡単に利用できる。

- void loop(){
- // 素晴らしいコードのアイデアをここで試す。
- return;
- // 残りのうまく動かないスケッチ
- // この部分のコードは実行されない。
- }

処理分岐だけど賛否両論ある関数

- goto

goto

- プログラム内のラベルに制御を移す。
- label:
- goto 識別子; // プログラムの制御をlabelに移す。

使用例

```
for(byte r = 0; r < 255; r++){
    for(byte g = 255; g > -1; g--){
        for(byte b = 0; b < 255; b++){
            if (analogRead(0) > 250){ goto bailout;}
            // more statements ...
        }
    }
}
bailout:
```

ヒント

- C言語ではgoto文の使用は推奨されていない。C言語関連の書籍の著者の中には、goto文は不要であると主張する人もいる。しかし、適切に使えば、ある種のプログラムを簡潔に記述することができる。多くのプログラマがgoto文の仕様に難色を示す理由は、goto文の乱用により、予期しないプログラムのフローを容易に作り出し、デバグができなくなることにある。
- とはいうものの、goto文が重宝する状況はあり、コーディングを簡易にする。一つの状況は、深いループやifブロックから特定の条件の場合に抜け出すことである。

- こんな感じで処理分岐をするためには多くの種類の関数があり、それぞれの特徴を生かして使いこなす必要がある。
- 使っていくうちに慣れると思うし、マイコンカーでもよく使われる。
- if、switch、breakはよく使われるので覚えておいて損はない。
- if文は基本中の基本である。

最後に

- 処理分岐を使わないと今後プログラムを書く際に思い通りの処理ができない。
- 次回はタイマー（時間の関数）を扱う。

特別課題

- サイトに特別課題1と2を上げてあるので作成して実行させること

課題

- スイッチを押したときにLEDを点灯させるプログラムを作る。
- スイッチとLEDは3つ。
- 抵抗は100Ω。