プログラミングサークル

C言語　for 1年生

第11回

「オセロプログラム②」

2022/7/1

前回のおさらい

前回までに以下の③までの処理が完了している。

①盤面をリセットし、初期石を置く
②標準入力で石を置く座標を入力する
③盤内かつすでに石が置かれていないか確認
④縦横斜め方向に1マスずつ確認
⑤敵色を挟んで自色があった場合はそれまでの石を自色に変える
⑥続行可能か判断する
⑦続行不可になればマスの数を数える

到達目標

今回の到達目標は以下の通りである。

「オセロの判別の仕組みを理解する」

それではやっていこう

オセロって意外と複雑

今回は、下の④～⑦のプログラムを考えていく。

①盤面をリセットし、初期石を置く

②標準入力で石を置く座標を入力する

③盤内かつすでに石が置かれていないか確認

④縦横斜め方向に1マスずつ確認

⑤敵色を挟んで自色があった場合はそれまでの石を自色に変える

⑥続行可能か判断する

⑦続行不可になればマスの数を数える

プログラム④-⑤-1

```
1   else{
2       flag=put(i, j,nowcolor2);
3       if (flag == 0) {
4           break;
5       }else if (flag == 1) {
6           printf("選択した場所には置けません\n");
7       }
8   }
```

・指定した場所が盤内で、なおかつ石がまだ置かれていない場合、
put**関数**を実行する。引数には指定した場所の座標と、現在の色を指定。
・戻り値は「flag」となっている。flag が1となってしまった場合、選択した場所には石を置けないことを示す。

プログラム④-⑤-2

```
1    int put(int x, int y,int nowcolor3) {
2        int i,j,n,k;
3        int flag=1;
```

このページ以降は、put関数の中身について述べる。

・仮引数x,yは、指定した座標（xが行、yが列）を表している。

・仮引数nowcolor3は、現在の石の色を表している。

・変数 i , j は、縦横斜め8方向の内どの方向を確認するかを定めるために必要である。

　例：i =−1 , j =1のとき、左斜め上を指定

・変数 n,k については後のページで述べる。

・変数flagは、1のとき石が置けない状態、0のとき石が置ける状態を表す。

プログラム④-1

- このページ以降は、put関数の中身である。
- 変数x,yは、指定した座標（xが行、yが列）を表している。
- 変数ｉ,ｊは、縦横斜め8方向の内どの方向を確認するかを定めるために必要である。
  例：i =−1 , j =1のとき、左斜め上を指定
- 5・6行目は、どの方向も指定していないので、continue文でスキップしている。
- 9・10行目は、隣が自色である方向はfor文をスキップする処理である。

```
1   for (i = -1; i <= 1; i++) {
2           for (j = -1; j <= 1; j++) {
3
4               /*置いたところは無視*/
5               if (i == 0 && j == 0) {
6                   continue;
7               }
8               /*隣が自色は無視*/
9               if (color[i+x][j+y]==nowcolor3) {
10                  continue;
11              }
```

プログラム④-2

```
1    /*各方向に1マスずつ確認*/
2                for (k = 1; k < 8; k++) {
3
4                    /*盤内だけでチェック*/
5                    if (x + i * k >= 0 && x + i * k < 8 && y + j * k >= 0 && y + j * k < 8) {
```

・2行目は、いくつ先の石で敵色の石を挟むかを、
　1マスずつ確認する処理である。
・5行目は、盤内の話だけで完結するために必要な処理である。
　例：x=1,i=−1,k=2のとき、盤外に出てしまうのでチェックしない

プログラム⑤-1

・2～4行目は、調べたい方向に自色が現れる前に空白があった場合、for文（kのループ）をスキップする処理である。

・6行目以降は、調べたい方向に自色が現れた場合、初めて置きたい場所に石を置くよう指示する（8行目）。
・そのうえで、挟んだ敵色の石を自色に変えている（9～11行目）。

・石が置ける場合に限り、変数flagの値を0にする。最終的にflagを戻り値とする。

```
1   /*空白があった場合はそれ以上試行する必要なし*/
2   if (color[x + i * k][y + j * k] == 0) {
3       break;
4   }
5   /*自色を見つけたらそこまでのコマをひっくり返す*/
6   else if (color[x + i * k][y + j * k] == nowcolor3) {
7       /*コマを置き自色に変える*/
8       color[x][y] = nowcolor3;
9       for (n = 1; n <= k; n++) {
10          color[x + i * n][y + j * n] = nowcolor3;
11      }
12      flag=0;
13      break;
14  }
```

プログラム⑥-1

前回調整として行った続行判断であったが、
本格的なオセロとするためにcheck関数を
書いていく。
右のコードはmain関数内の処理である。

```
1   /*続行判断*/
2   badpoint=check(badpoint,nowcolor);
3   if(badpoint==2){
4       break;
5   }
```

・引数や戻り値に書かれている変数badpointに
ついてはこれから述べていく。
・現在の色を表す引数nowcolorも引数とする。

・badpointは初めは0であるが、2となった場合
（両者とも駒が置けない状態となった場合）は、
main関数のwhile文からbreakで抜け出す。

プログラム⑥-2

```
1   int check(int badpoint,int nowcolor2){
2       int k,i,j,x,y,flag=1;
3       for (x = 0; x < 8; x++) {
4           for (y = 0; y < 8; y++) {
5               for (i = -1; i <= 1; i++) {
6                   for (j = -1; j <= 1; j++) {
```

このページ以降は、**check関数**の中身について述べる。

・仮引数の意味は前ページの実引数と変わらない。
・変数k,i j,x,y については、**put関数**と意味は全く同じである。
x,y については、ここでは盤内すべての座標について調べている。
・変数flagは初期値を1としている。このまま1であった場合、その色では続行不可、0になった場合はその色で続行可であることを示す。

プログラム⑥-3

盤上で空白があった場合に、そのマスが本当における場所であるのかを判断する。

・1～3行目では、石が置かれている場合はその座標では続行判断をする意味がないのでcontinue文でスキップしている。
・4行目で、念のためにflagを1としている。

・7～9行目、11行目～13行目の処理は**プログラム④-1**と意味は同じ。

```
1   if (color[x][y] == 1 || color[x][y] == 2) {
2       continue;
3   }
4   flag = 1;
5
6   /*置いたところは無視*/
7   if (i == 0 && j == 0) {
8       continue;
9   }
10  /*隣が自色は無視*/
11  if (color[i+x][j+y]==nowcolor2) {
12      continue;
13  }
```

プログラム⑥-4

```
1   /*各方向に1マスずつ確認*/
2   for (k = 1; k < 8; k++) {
3
4       /*盤内だけでチェック*/
5       if (x + i * k >= 0 && x + i * k < 8 && y + j * k >= 0 && y + j * k < 8) {
6
7           /*空白があった場合はそれ以上試行する必要なし*/
8           if (color[x + i * k][y + j * k] == 0) {
9               break;
10          }
11          /*自色を見つけたらそこまでのコマをひっくり返す*/
12          else if (color[x + i * k][y + j * k] == nowcolor2) {
13              flag=0;
14              goto end;
15          }
```

・2行目～10行目までの処理は**プログラム④-2・⑤-1**と意味は同じ。

・ある空白で**石が置ける場合**に限り、変数flagの値を0にする。
・14行目の「**goto end**」のような文を**goto文**という。gotoと書いた後にラベル名を指定し、次に任意の場所にそのラベル（名前は自由）を書く。そのようにすると、goto文の次はそのラベルの位置が次に実行される文になる。

プログラム⑥-5

goto文を用いることで、xのforループ～kのforループまで
5つのforループから抜け出すことができる。

・さきほどラベル名としてendを指定した。**goto文**の後は
右のコードの1行目に飛ぶ。
・石が置けない場合には変数**badpoint**を1に、石が置ける
場合には**badpoint**の値を0にリセットする。
・7行目では、念のために**flag**を1としている。
・最終的に**badpoint**を戻り値とする。

```
1    end:
2    if(flag == 1) {
3        badpoint+=1;
4    }else if(flag==0){
5        badpoint=0;
6    }
7    flag==1;
8    return badpoint;
```

**※goto文は使うと複雑なバグを引き起こすこともあるので
原則使わないようにする**

プログラム⑥-6

右のコードはmain関数内で石を置く処理である。

```
1   /*始める*/
2   if(badpoint!=0){
3       printf("置ける場所はありません");
4       sleep(3);
5   }else{
6       play(nowcolor);
7   }
```

・check関数で得られた（最初を除く）badpointの値が0ではない場合（基本的には値が1の場合）は石は置けない状態であることを出力する。

・4行目ではsleep関数を用いている。
引数で指定した秒数をスリープする。
・プログラムの先頭に #include <stdlib.h> が必要
・ここでは3秒スリープさせることで、急に次のターンに行くことを防止している。

sleep関数
sleep(秒数);

・変数badpointの値が0である場合は、play関数を実行する。

プログラム⑦-1

右のコードは、続行不可になった時にマスの数を数えるfinish**関数**である。

・4～13行目で各マスをカウントして、
　白と黒の石数を調べている。

・15～25行目でその勝敗を表示している。

```
1   void finish() {
2       int blackcount = 0;
3       int whitecount = 0;
4       for (int i = 0; i < 8; i++) {
5           for (int j = 0; j < 8; j++) {
6               if (color[i][j] == 1) {
7                   whitecount = whitecount + 1;
8               }
9               else if (color[i][j] == 2) {
10                  blackcount = blackcount + 1;
11              }
12          }
13      }
14
15      printf("黒:%d 白:%d\n", blackcount, whitecount);
16
17      if (blackcount > whitecount) {
18          printf("黒の勝ち\n");
19      }
20      else if (blackcount < whitecount) {
21          printf("白の勝ち\n");
22      }
23      else if (blackcount == whitecount) {
24          printf("引き分け\n");
25      }
26  }
```

プログラム例

ここまでの全内容を加えたうえで、最終的に
main関数の中身は右のようになる。

```c
int main() {
    int badpoint,nowcolor;
    system("cls"); /*初手*/
    reset(); /*盤をリセット*/
    display(); /*描写*/
    badpoint=0; /*最初はどこも置ける*/
    nowcolor=2; /*最初は黒*/

    while (1){
        if (nowcolor == 2) {
            printf("黒のターンです\n");
        }else if (nowcolor == 1) {
            printf("白のターンです\n");
        }

        /*始める*/
        if(badpoint!=0){
            printf("置ける場所はありません");
            sleep(3);
        }else{
            play(nowcolor);
        }

        /*描写*/
        system("cls");
        display();
        /*交替*/
        nowcolor=change(nowcolor);
        /*続行判断*/
        badpoint=check(badpoint,nowcolor);
        if(badpoint==2){
            break;
        }
    }
    finish();
}
```

プログラムを実行すると、
オセロプログラムの完成！！

osero(vsUSER).txt

おわり

※何らかのエラーが出たら報告してほしいです！

みなさんここまで学習順調でしょうか？
細かく解説しているつもりですが、何か質問があればチャットください～

学習できていない方は、とりあえず実行するだけでもいいと思います
大学での授業など、学習が追い付いたら理解しながら作ってみてください～

次回は発展的な内容として、オセロ（VS単純AI版）を作ります！