ソフトウェア設計及び実験

6119019056 山口力也

2019/04/23 日提出

1　プログラミング作法について

私は高専時代に卒業研究で深層学習について研究していた. その時初めてPython を学び, プログラミング作法の大切さを痛感した.Python を学ぶ前もある程度インデントくらいはしていたが, 編集している途中にずれることがよくあった. しかし, それまでに触った C や Java,Fortran といった言語ではインデントがずれていてもプログラム自体は動いてしまっていたので, 特に気にすることもなかった.Python ではインデントが統一されており, 私のように変なインデントをしているとプログラムが動かない. そのため卒業研究の序盤はそこでかなり苦しみ, 同期のみなが研究に勤しんでいる中自分はプログラミングの基礎中の基礎のインデントをもう一度復讐していた. 結果的にはインデントは vim のコマンドで”gg=G”と打ってしまえばほぼすべて解決したが, インデントに限らず変数の定義の仕方など, 担当教員から何度も注意され, そこでプログラミング作法の大切さに気付いたと思う. しかし, ＝の前にスペースを入れるべきか否か,if 文の”{”は改行するべきか否か, など人によって意見が異なるプログラミング作法もあると思う. 実際, 自分の周りでも意見が割れていた. そういったものに関してはその人の”流儀”として受け入れるべきなのかもしれない. みながみな同じようにコードを書いていれば理解しやすく効率的だが, 現実問題それは不可能だ. もちろんプログラミング作法に忠実であることは大切であるが, 絶対にこの書き方じゃないとダメ!ではなく, 他の人の書き方も受け入れる柔軟性も必要だと思う. 長々と書いてしまったが, 結局言いたいのはプログラミング作法について学ぶことは重要であるということである. プログラミング作法を学べば, 単にコードが綺麗になるだけでなく, 様々なことが学べると思う. 自分は適当なエディタで適当に Tab キーを押していたためインデントに苦しんだが, その解決策として vim という最高のテキストエディタに巡り会えた. 他の人だったらもっと良い発見があるかもしれない. とは言っても, 自分はまだまだ初心者で他の人のコードを見ると自分のコードの汚さと無駄の多さを感じる. これからもプログラミング作法に注意を払い, グループ開発などで誰が見ても理解できるようなわかりやすいコーディングを心掛けたい.

2　プログラミング課題

2.1　課題 1 について

サンプルプログラム sample_prog.c を、斜めでも挟めるように修正した. 追加した部分のソースコードを以下ソースコード 1 に示す.

ソースコード 1: 課題 1

```
1  void zero_to_one(int karival[][MAX_COL],int nval[][MAX_COL]){
2    for(int i=0; i<MAX_ROW; i++){
3      for(int j=0; j<MAX_COL; j++){
4        nval[i][j] = karival[i][j];//nval に karival を代入
5      }
6    }
7
8    for(int i=0; i<MAX_ROW; i++){
9      for (int j=0; j<MAX_COL; j++){
10       if ( karival[i][j] == 0 && j-1 >= 0 && j+1 <MAX_COL
             && i-1 >=0 && i+1 < MAX_ROW ){//縦の端じゃないかつ
             横の端じゃないとき
11         if( ( karival[i-1][j-1] + karival[i+1][j+1] ) == 2
               || ( karival[i-1][j+1] + karival[i+1][j-1] ) ==
                2 ){//斜めで挟んだ時
12           nval[i][j] = 1;
13         }
14       }
15     }
16   }
17 }
```

縦の端と横の端は斜めで挟まれることはないので,if 文を用いて条件から排除し, 斜めで挟まれる場合は現在見ている配列の要素が 0 でかつ, 現在見ている配列から縦横ともに+1 された要素と縦横ともに-1 された要素が 1 である場合, または縦を-1 横を+1 した要素と縦を+1, 横を-1 した要素が 1 である場合であるのでそのようにコードを書き換えた. 工夫した点は,0 を 1 に変換するところを関数化し, 呼び出しを簡単にしたところである. 動作としては斜めの部分も 1 となり正常に動作した.

2.2　課題 2 について

サンプルプログラム sample_prog.c で, 縦横両方に挟まれている時に 0 を 1 に変換するように修正した. 変更した部分のソースコードを以下ソースコード 2 に示す.

ソースコード 2: 課題 2

```
1  void zero_to_one(int karival[][MAX_COL],int nval[][MAX_COL]){
2    for(int i=0; i<MAX_ROW; i++){
3      for(int j=0; j<MAX_COL; j++){
4        nval[i][j] = karival[i][j];
5      }
6    }
7
8    for(int i=0; i<MAX_ROW; i++){
9      for (int j=0; j<MAX_COL; j++){
10       if ( karival[i][j] == 0 && j-1 >= 0 && j+1 <MAX_COL
             && i-1 >=0 && i+1 < MAX_ROW ){//縦の端じゃないかつ
             横の端じゃないとき
11         if( ( karival[i-1][j] +karival[i+1][j] ) == 2 && (
               karival[i][j-1] + karival[i][j+1] ) == 2 ){
12           nval[i][j] = 1;
13         }
14       }
15     }
16   }
```

縦横両方に挟まれるということは現在の要素の上と下の要素を足せば 2 になり, 同様に左右の要素を足せば 2 になる時であるのでそのようにプログラムを変更した. こちらもソースコード 1 と同様に関数化した.

2.3 課題 3 について

サンプルプログラム sample_prog.c で, 最終的に 1 で挟まれている 0 がないように修正した. これは 2.1 節において斜めで挟んだ時にできた”1”が新たに縦や横や斜めで挟む要素になるので, その場合はもう一度挟まれている 0 を 1 に変換するという意味である. 以下に作成した部分のソースコードを以下ソースコード 3 に示す.

ソースコード 3: 課題 3

```
1  int zero_to_one(int karival[][MAX_COL],int nval[][MAX_COL]){
2    int flag = 0;
3    for(int i=0; i<MAX_ROW; i++){
4      for(int j=0; j<MAX_COL; j++){
5        nval[i][j] = karival[i][j];
6      }
7    }
8    for(int i=0; i<MAX_ROW; i++){
9      for (int j=0; j<MAX_COL; j++){
```

```
10        if ( karival[i][j] == 0 && j-1 >= 0 && j+1 < MAX_COL
              ){
11          if ( ( karival[i][j-1] + karival[i][j+1] ) == 2
                ){//横で挟んだとき
12            nval[i][j] = 1;
13            flag++;
14          }
15        }
16        if ( karival[i][j] == 0 && i-1 >= 0 && i+1 < MAX_ROW
              ){
17          if ( ( karival[i-1][j] + karival[i+1][j] ) == 2
                ){//縦で挟んだとき
18            nval[i][j] = 1;
19            flag++;
20          }
21          if ( karival[i][j] == 0 && j-1 >= 0 && j+1 <MAX_COL
                && i-1 >=0 && i+1 < MAX_ROW ){//縦の端じゃない
               かつ横の端じゃないとき
22            if( ( karival[i-1][j-1] + karival[i+1][j+1] ) ==
                  2 || ( karival[i-1][j+1] + karival[i+1][j-1]
                  ) == 2 ){//斜めで挟んだとき
23              nval[i][j] = 1;
24              flag++;
25            }
26          }
27        }
28      }
29    }
30    return flag;
31  }
32  //~~~~~~中略~~~~~~
33
34  int main{int argc, char** argv){
35    //~~~~~~~~中略~~~~~~~~~
36    // 1に挟まれた 0を 1に変換する処理
37    // 更新後の数値は別の配列に保存
38    while(1){
39      int flag = 0;//変換する必要があるかないかの判定用
40      flag = zero_to_one(val,nval);//0を 1に変換する関数
41      printf("挟まれていた数は%d\n",flag);
42      FILE* wfp = fopen(opath, "w");
43      if( wfp == NULL) {
44        printf("%s file not open!\n", opath);
45        return -1;
46      }
47      for( i=0; i<MAX_ROW; i++){
```

```
48        for( j=0; j<MAX_COL; j++){
49          printf("%d,", nval[i][j]);
50          fprintf(wfp, "%d", nval[i][j]);
51        }
52        printf("\n");
53        fprintf(wfp, "\n");
54      }
55      fclose(wfp);
56      if(flag==0){
57        break;
58      }
59      for( i=0; i<MAX_ROW; i++){
60        for( j=0; j<MAX_COL; j++){
61          val[i][j] = nval[i][j];
62        }
63      }
64    }
65    return 0;
66  }
```

flag を作り, 変換するたびに flag を足していき,flag が 0 になった時無限ループを抜けるというプログラムに修正した. 関数化はできたが, 個人的には無駄が多いプログラムになってしまったと感じている. ループの中で毎回全ての配列の要素を確認しているので処理が重い.while 文の中ももう少し簡潔に分かりやすく書けると思う. 今回は自分の力と時間との兼ね合いでチャレンジできなかったが, 機会があればまたチャレンジしたい.