プログラミング演習I第11回レポート

学籍番号：2364902

名前：キム　ギュソク

(1) 課題番号と課題内容

A. 基本課題２

i. 下記のdataは、迷路を表している。8:壁 0:通路 1:スタート 9:ゴール である時、スタートからゴールに辿り つく全ての経路とゴール到達までの距離(コマ数)を出力しなさ。

```
#define N 10
int data[N][N]={
  {8,8,8,8,8,8,8,8,8,8},
  {8,1,0,0,8,8,0,0,0,8},
  {8,0,8,0,0,0,0,8,0,8},
  {8,0,8,8,8,0,8,8,0,8},
  {8,0,0,8,8,0,0,8,0,8},
  {8,8,0,0,8,0,8,8,0,8},
  {8,0,8,0,8,0,0,8,0,8},
  {8,0,0,0,8,8,0,0,0,8},
  {8,8,8,0,0,0,0,8,0,8},
  {8,8,8,8,8,8,8,8,9,8}};
```

(2) フローチャートまたは疑似言語によるアルゴリズムの記述

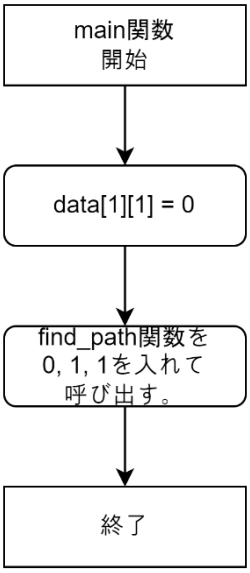


図 1。main関数フローチャート

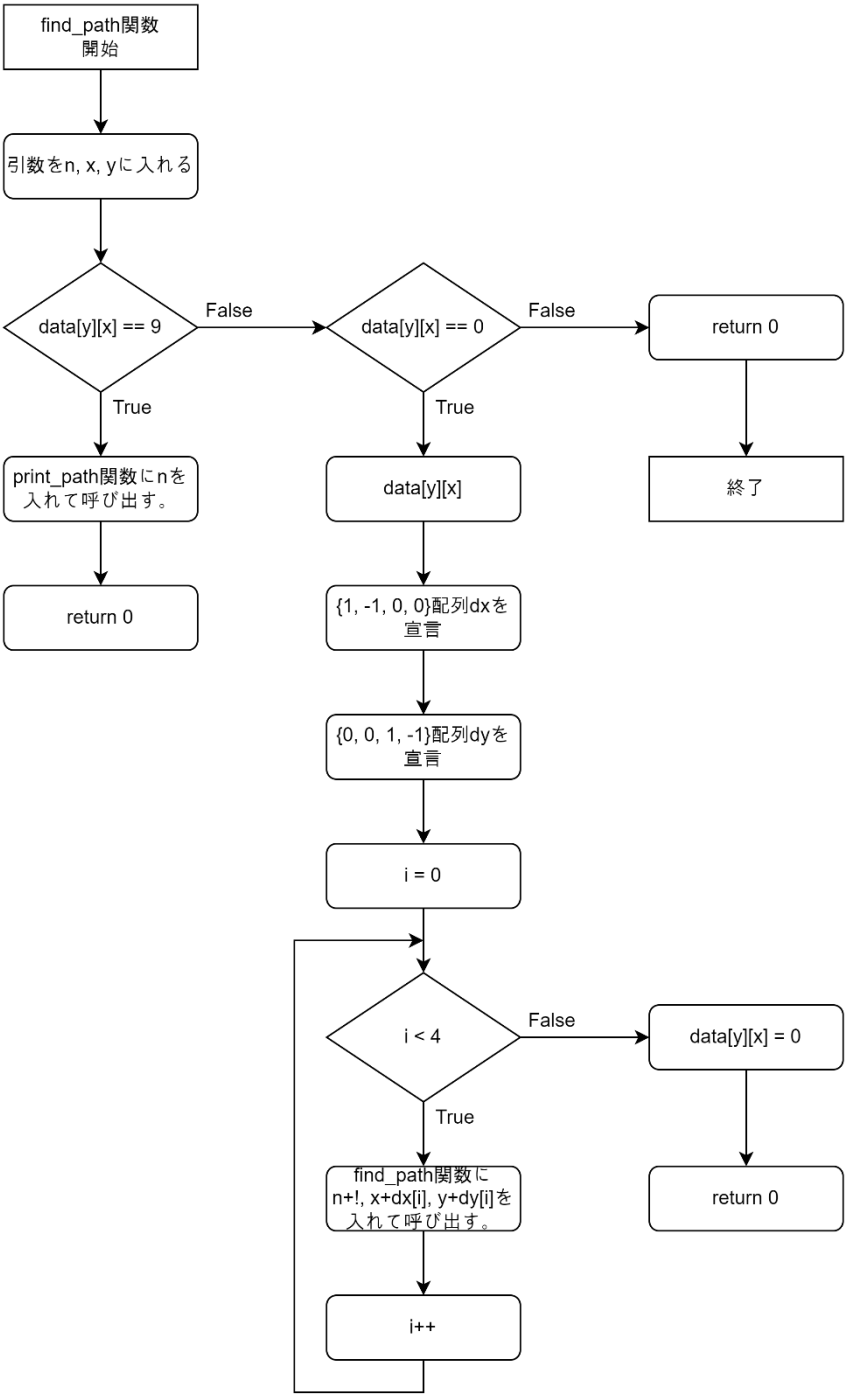


図 2。find_path関数フローチャート

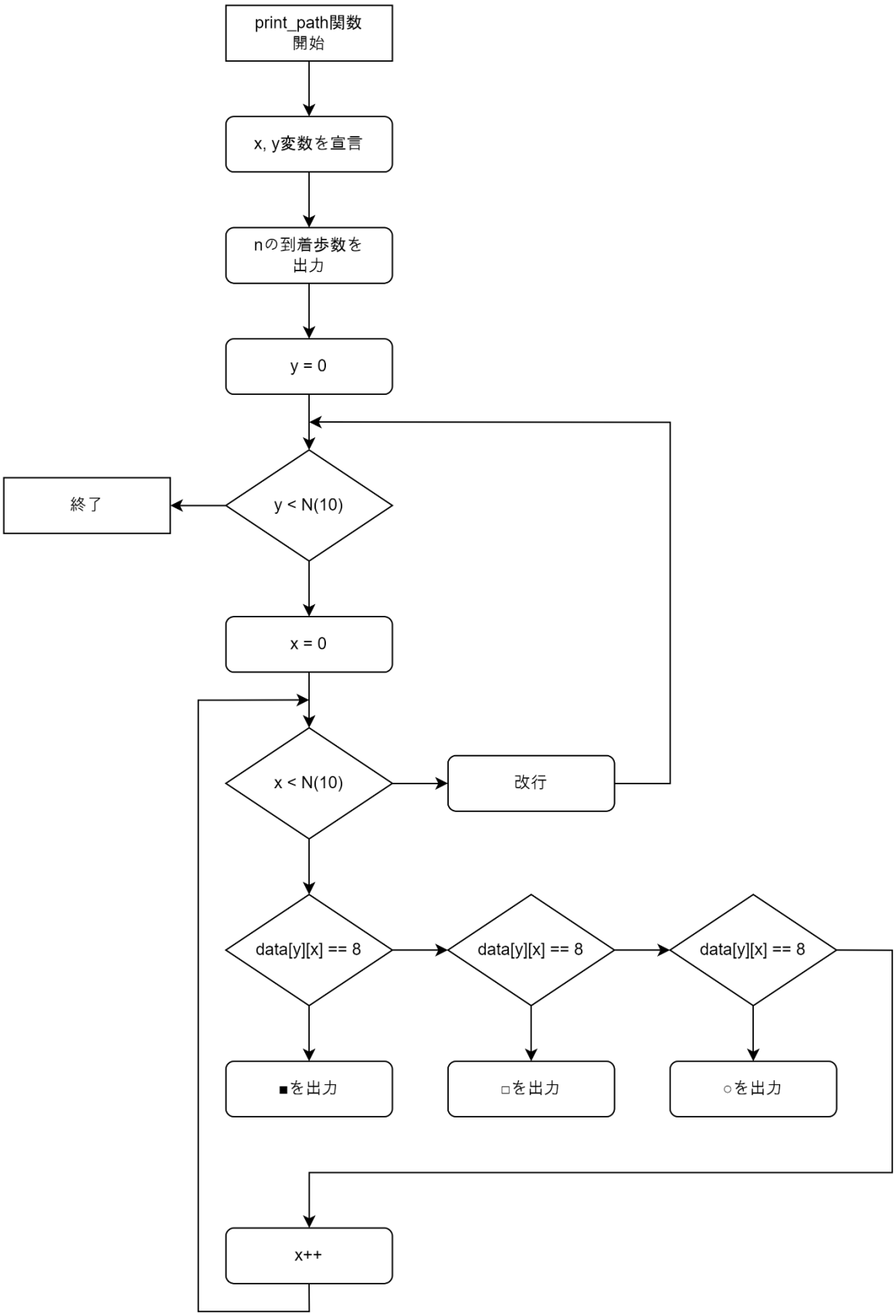


図 3。printf_path関数フローチャート

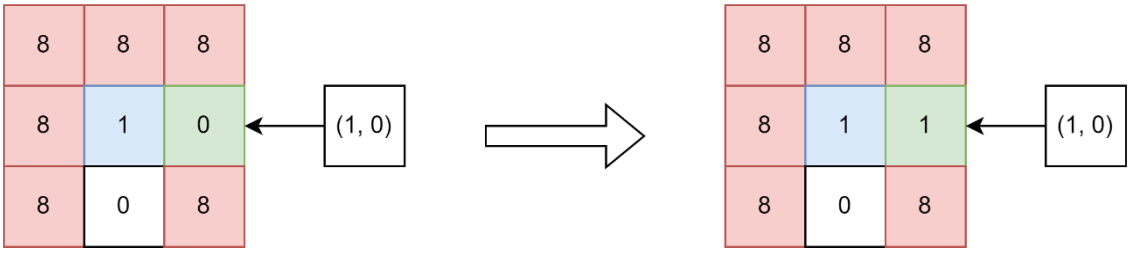
(3) アルゴリズムが正しいことの説明

A. 正当性

i. (x, y)を(1, 0), (-1, 0), (0, 1), (0, -1)順に再帰呼び出しで全ての方向に移動して現在の位置の要素が何かを調べる。例えば、ゴールである9の場合mapを出力して 0 をreturnすることで他の経路を探しに行く。そして、現在の位置が壁である 8 の場合何もせず 0 をreturnする。そして、現在の位置が通路である 0 の時は現在の位置を 1 にして通ったことを示す。その後、再帰呼び出しをしてまた次の通路とゴールを探す。これをゴールを見つけるまでに続けることで全探索アルゴリズムで経路を見つけることができる。このような過程を図で見ると

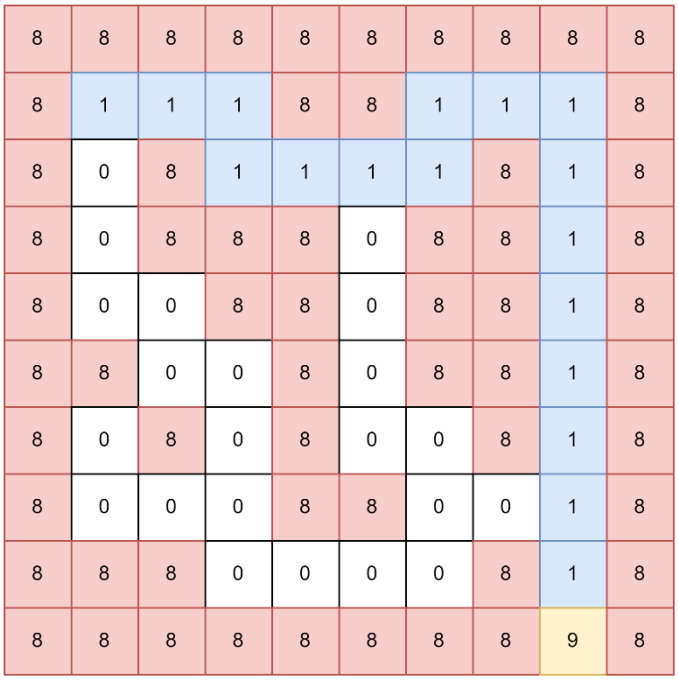
8	8	8	8	8	8	8	8	8	8
8	1	0	0	8	8	0	0	0	8
8	0	8	0	0	0	0	8	0	8
8	0	8	8	8	0	8	8	0	8
8	0	0	8	8	0	8	8	0	8
8	8	0	0	8	0	8	8	0	8
8	0	8	0	8	0	0	8	0	8
8	0	0	0	8	8	0	0	0	8
8	8	8	0	0	0	0	8	0	8
8	8	8	8	8	8	8	8	9	8

このようなmapで 1 の位置から始まるので(1, 0), (-1, 0), (0, 1), (0, -1)順に調べると



繰り返しの(1, 0)で通路を見つけて 1 に変えて通ったことを示すこ

れを全ての通路を見つけながらゴールを見つけた時にmapを出力することで正しく経路を見つけることができる。



B. 停止性

i. このプログラムでは無限ループに入る可能性を持っているループはfind_path関数の再帰関数とprint_path関数の二重ループである。

1. find_path関数

A. ゴールを見つけた時にreturn 0されて次の段階であるback trackingを始まるので他の経路を見つけるまでに通った通路を 0 に戻す。そして、全ての通路を通った後にはスタート地点に戻って関数がreturn 0されて終了するので再帰が終了する。よって、全ての経路を探索して終わる正しく停止する再帰関数である。

2. print_path関数

A. この関数はmapデータを出力するだけなので10×10のmapデータの列の繰り返しと行の繰り返しであるため二重ループ全てi, j < 10とおいてループするので正しく10×10のmapを出力して停止する。

(4) ソース・プログラムの説明

A. ソースコードの説明

i. stdio.hライブラリをincludeしてmapデータの行と列の長さであるNを#defineで10として宣言する。そして、mapデータを2重配列で宣言する。そして、全ての関数のプロトタイプを宣言してmain関数でスタート地点を 0 に初期化してから経路を探すfind_path関数をn ＝ 0とスタート地点(1, 1)を引数に入れて呼び出してmain関数は終了する。find_path関数では再起関数であるため終了条件としてdata[y][x]が 9 （ゴール）である時に終了するようにreturn 0 をする。そして、ゴールではない時に次の現在の位置に通った印として 1 を設定して次の位置に移るためのx方向移動配列とy方向移動配列を(1, 0), (-1, 0), (0, 1), (0, -1)で作る。その後、全ての方向を調べるために4回繰り返して次の位置を入れて再帰する。再帰が終わったとき（ゴールや詰まったとき）に次の経路を探索するためのback trackingとして再帰の過程で戻りながら現在の位置を 0 に変える。そして、return 0 する。そして、mapをprintするための関数print_pathで歩数を表すnを出力して、繰り返しを利用して壁は■通路は□それ以外は通った意味として○を出力する。

(5) 考察

A. このアルゴリズムでは毎回現在の位置で全ての方向に行けるか調べる。しかし、多くの経路が進める方向にずっと進傾向があると考えられる。例えば、今回の課題の例を見ると最後にゴールまで至るときにずっと下に下ることがわかる。このように普遍的な特徴ではないがずっと同じ方向に通路がある時に無駄な計算が生じる。これを前回に通路があった方向を次の再帰呼び出すときに渡すことで同じ方向にずっと通路がある時に無駄な計算はスキップしていけなかったときに全ての方向を調べる方法がより効率的であると考えられる。また、考えられる方法としては全ての方向から進めて調べることが時間的に一番効率的であると考える。回路の考え方で直列回路より並列回路の方がより時間的な効率を持っていることを考えるとわかると思う。また、並列的に探索することは4人が各自の方向から探索することと同じことなので時間的に遥かに効率的

である。

(6) 感想

A. 探索アルゴリズムを作ることで繰り返し文と再帰関数の違いをはっきりわかるようになった。また、探索アルゴリズムの効率性について考えるようになり、より効率的なアルゴリズムについて考えるようになった。