上級プログラミング第四回レポート課題

2019/06/28

us162039　梶田悠

2019

目次

[レポートの課題内容 1](#_Toc10909677)

[補足 2](#_Toc10909678)

[コード 2](#_Toc10909679)

[実行結果 7](#_Toc10909680)

[考察 7](#_Toc10909681)

[不適切なオプションで実行された場合について 7](#_Toc10909682)

[デバックの効率化 8](#_Toc10909683)

# レポートの課題内容

２次元の迷路をキューとスタックのデータ構造を利用して解いてみよう．地図情報は n × n の２次元配列で与えられ，スタートを点 (0, 0)，ゴールを点 (n − 1, n − 1) とする．配列の各セルに「-1」（壁）と「0」（通過可能）の初期値が入っている．問題 1–3 に回答せよ．問題 3 は発展問題とする．（各問の詳細な問題文は割愛）

## 補足

コードは[github](https://github.com/HarukaKajita/jpro_repot04)にアップロードしてあるので、レポートのコードが読みづらい場合は適宜参照して下さい。

# コード

ソースコード全文を以下に掲載する。

コード Point.h

|  |
| --- |
| #pragma once  #include <iostream>  using namespace std;  #pragma once  class Point {  private:  int x, y;  void addX(int x) { this->x += x; };  void addY(int y) { this->y += y; };  public:  Point(int \_x = -1, int \_y = -1) {  x = \_x;  y = \_y;  }  int getX() {  return x;  }  int getY() {  return y;  }  Point operator+(Point);  void operator+=(Point);  };  ostream& operator<<(ostream&, Point); |

コード Point.cpp

|  |
| --- |
| #pragma once  #include "Point.h"  ostream& operator<<(ostream& stream, Point p) {  stream << "(" << p.getX() << ", " << p.getY() << ")";  return stream;  }  Point Point::operator+(Point p) {  return Point(this->getX() + p.getX(), this->getY() + p.getY());  }  void Point::operator+=(Point p) {  this->addX(p.getX());  this->addY(p.getY());  } |

コード 　Node.h

|  |
| --- |
| #pragma once  #include <iostream>  using namespace std;  #include "Point.h"  //Node  class Node {  private:  Point data;  Node\* next;  public:  Node(Point p, Node\* n) {  data = p;  next = n;  }  //accessor  Node\* getNext() { return next; }  void setNext(Node\* np) { next = np; }  Point getData() { return data; }  }; |

コード 　Queue.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <cstdlib>  using namespace std;  #include "Node.h"  class Queue {  private:  Node\* front;  Node\* rear;  public:  Queue() { front = rear = NULL; }  ~Queue();  void enqueue(Point);  Point dequeue();  bool isEmpty();  }; |

コード 5 Queue.cpp

|  |
| --- |
| #pragma once  #include "Queue.h"  void Queue::enqueue(Point p){  //新しいノードを作成  //リアに追加し、リアを更新  Node\* newNode = new Node(p, NULL);  bool emptyFlag = isEmpty();  if(emptyFlag)front = newNode;  if(!emptyFlag)rear->setNext(newNode);  rear = newNode;  #ifdef \_DEBUG  cout << "enqueue : " << newNode << " next : " << newNode->getNext() << endl;  #endif  }  Point Queue::dequeue(){  //先頭からデータを吐き出し  //ノードを削除  //先頭を更新  if(isEmpty()){  cerr << "Error : front is NULL. can't dequeu" << endl;  exit(EXIT\_FAILURE);  }  Point data = front->getData();  Node\* tmp = front;  front = front->getNext();  #ifdef \_DEBUG  cout << "dequeue : " << tmp << endl;  cout << "front : " << front << endl;  #endif  delete tmp;  return data;  }  bool Queue::isEmpty(){  return front == NULL;  }  Queue::~Queue(){  while(!isEmpty())  dequeue();  } |

コード Stack.h

|  |
| --- |
| #pragma once  #include "Node.h"  class Stack {  private:  Node\* head;  public:  Stack() { head = NULL; }  ~Stack() { while (!isEmpty())pop(); }  void push(Point);  Point pop();  bool isEmpty() { return head == NULL; }  }; |

コード Stack.cpp

|  |
| --- |
| #pragma once  #include "Stack.h"  void Stack::push(Point s) {  Node\* newHead = new Node(s, head);  head = newHead;  }  Point Stack::pop() {  if (head == NULL) {  cerr << "StackError : error can't pop " << endl;  exit(EXIT\_FAILURE);  }  Point ret = head->getData();  Node\* ptr = head->getNext();  delete head;  head = ptr;  return ret;  } |

コード jpro\_report04.cpp

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <cstdlib>  using namespace std;  //#define Q1  #define Q2  //#define Q3  // Node, Queueと Stack クラスはここに定義  // または include してもよい  #include "Queue.h"  #include "Stack.h"  // 迷路に設置するセルの属性  enum {  WALL = -1, // 壁  EMPTY = 0, // 通路  ROUTE = 1, // ゴールまでの道  LEFT = 1001, // 左のセルからきた（探索途中用）  RIGHT, // 右のセルからきた（探索途中用）  UP, // 上のセルからきた（探索途中用）  DOWN, // 下のセルからきた（探索途中用）  START // 出発セル（探索途中用）  };  bool solveByQueue(int\*\* arr, const int n) {  // 空のキューを用意する  Queue q;  // 始点(0,0)をキューに追加する  q.enqueue(Point(0, 0));  arr[0][0] = START;  // [問題１]幅優先探索を行う（キューを利用してコードを補完する）  while (!q.isEmpty()) {  Point p = q.dequeue();  #ifdef \_DEBUG  cout << p << "を中心に探索開始" << endl;  #endif // \_DEBUG  Point neighbors[4] = {  Point(p.getX() , p.getY() - 1),  Point(p.getX() + 1, p.getY()),  Point(p.getX() , p.getY() + 1),  Point(p.getX() - 1, p.getY())  };  for (int i = 0; i < 4; i++) {  const int x = neighbors[i].getX();  const int y = neighbors[i].getY();  if (x == n || x == -1 || y == n || y == -1) continue;  if (arr[x][y] == EMPTY) {  q.enqueue(neighbors[i]);  #ifdef \_DEBUG  cout << neighbors[i] << "をエンキューしました" << endl;  #endif // \_DEBUG  #ifdef Q1  arr[x][y] = ROUTE;  #elif defined Q2 || defined Q3  if (i == 0) {  arr[x][y] = RIGHT;  #ifdef \_DEBUG  cout << "RIGHT" << endl;  #endif // \_DEBUG  }  else if (i == 1) {  arr[x][y] = UP;  #ifdef \_DEBUG  cout << "UP" << endl;  #endif // \_DEBUG  }  else if (i == 2) {  arr[x][y] = LEFT;  #ifdef \_DEBUG  cout << "LEFT" << endl;  #endif // \_DEBUG  }  else if (i == 3) {  arr[x][y] = DOWN;  #ifdef \_DEBUG  cout << "DOWN" << endl;  #endif // \_DEBUG  }  #endif  }  }  }  // ゴールセルの状態を確認する  if (arr[n - 1][n - 1] == EMPTY)  return false;  return true;  }  //Stackを利用して迷路を解く  bool solveByStack(int\*\* arr, const int n) {  arr[0][0] = START;  Stack s;  s.push(Point(0, 0));  while (!s.isEmpty()) {  Point p = s.pop();  Point neighbor[4] = {  Point(p.getX() + 1, p.getY()),  Point(p.getX() - 1, p.getY()),  Point(p.getX() , p.getY() - 1),  Point(p.getX() , p.getY() + 1)  };  for (int i = 0; i < 4; i++)  {  int x = neighbor[i].getX();  int y = neighbor[i].getY();  if (x == -1 || x == n || y == -1 || y == n) continue;  if (arr[x][y] == EMPTY) {  if (i == 0) {  arr[x][y] = UP;  }  else if (i == 1) {  arr[x][y] = DOWN;  }  else if (i == 2) {  arr[x][y] = RIGHT;  }  else if (i == 3) {  arr[x][y] = LEFT;  }  s.push(neighbor[i]);  }  }  }  // ゴールセルの状態を確認する  if (arr[n - 1][n - 1] == EMPTY)  return false;  return true;  }  // Mapを初期化する  void initMap(int\*\* arr, const int n) {  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  if (arr[i][j] != WALL)  arr[i][j] = EMPTY;  }  // Mapをプリントする  void printMap(int\*\* arr, const int n) {  for (int i = -1; i <= n; i++) {  for (int j = -1; j <= n; j++) {  if (i == -1 || i == n || j == -1 || j == n)  cout << "■";  else if (arr[i][j] == WALL)  cout << "×";  else if (arr[i][j] == ROUTE)  cout << "●";  #ifdef \_DEBUG  else if (arr[i][j] == UP)  cout << "↑";  else if (arr[i][j] == LEFT)  cout << "←";  else if (arr[i][j] == DOWN)  cout << "↓";  else if (arr[i][j] == RIGHT)  cout << "→";  #endif // \_DEBUG  else  cout << "　";  }  cout << endl;  }  }  void printPath(int\*\* arr, const int n) {  // [問題２]スタックを利用して，ゴールまでの経路と地図を描画する  // 描画する際に，printMap関数を利用してよい  Stack s;  Point p = Point(n - 1, n - 1);  s.push(p);  while (arr[p.getX()][p.getY()] != START) {//スタートまでの経路をスタックに格納する  int dir = arr[p.getX()][p.getY()];  #ifdef \_DEBUG  cout << p << " : ";  #endif // \_DEBUG  if (dir == UP) {  p += Point(-1, 0);  #ifdef \_DEBUG  cout << "UP" << endl;;  #endif // \_DEBUG  }  else if (dir == LEFT) {  p += Point(0, -1);  #ifdef \_DEBUG  cout << "LEFT" << endl;;  #endif // \_DEBUG  }  else if (dir == DOWN) {  p += Point(1, 0);  #ifdef \_DEBUG  cout << "DOWN" << endl;;  #endif // \_DEBUG  }  else if (dir == RIGHT) {  p += Point(0, 1);  #ifdef \_DEBUG  cout << "RIGHT" << endl;;  #endif // \_DEBUG  }  s.push(p);//経路をスタックに記録  }  //ゴールからスタートまでの座標がスタックに積まれた状態になっている  //経路を出力する  cout << "Path : ";  while (!s.isEmpty()) {  Point p = s.pop();  arr[p.getX()][p.getY()] = ROUTE;  cout << p << " ";  }  cout << endl;  printMap(arr, n);  }  int main() {  int n;  int\*\* ins;  // 問題入力の読み込み  char fName[100];  cout << "問題例ファイルを入力してください：";  cin >> fName;  ifstream fin(fName, ios::in);  if (!fin) {  cout << "Error: ファイル(" << fName << ")が開けません．" << endl;  exit(1);  }  fin >> n;  ins = new int\* [n];  for (int i = 0; i < n; i++) {  ins[i] = new int[n];  for (int j = 0; j < n; j++)  //※下方向が、X軸正の向き。  //※右方向が、Y軸正の向き。  fin >> ins[i][j];  }  // 入力地図の描画  printMap(ins, n);  // キューを利用する迷路探索  bool rst = solveByQueue(ins, n);  #ifdef \_DEBUG  printMap(ins, n);  #endif // \_DEBUG  if (rst) {  cout << "Queue: ゴールまで到達可能" << endl;  #if defined Q2 || defined Q3  printPath(ins, n);  #endif  }  else {  cout << "Queue: ゴールまで到達不可能" << endl;  }  #ifdef Q3  //[問題３] に関係するコード  // 入力地図の初期化  initMap(ins, n);  // スタックを利用する迷路探索  rst = solveByStack(ins, n);  if (rst) {  cout << "Stack: ゴールまで到達可能" << endl;  printPath(ins, n);  }  else {  cout << "Stack: ゴールまで到達不可能" << endl;  }  #endif // Q3  for (int i = 0; i < n; i++)  delete[] ins[i];  delete[] ins;  return 0;  } |

# 実行結果

## 問題１

|  |
| --- |
| 問題例ファイルを入力してください：maze1.txt  ■■■■■■  ■　×　　■  ■　　　×■  ■×　××■  ■　×　　■  ■■■■■■  Queue: ゴールまで到達不可能 |

|  |
| --- |
| 問題例ファイルを入力してください：maze2.txt  ■■■■■■■  ■　　××　■  ■　×　　　■  ■　　　×　■  ■××　×　■  ■　　　×　■  ■■■■■■■  Queue: ゴールまで到達可能 |

|  |
| --- |
| 問題例ファイルを入力してください：maze3.txt  ■■■■■■■■  ■　×　　　　■  ■　　　×　×■  ■×　××　×■  ■　　　　　　■  ■×　×　×　■  ■　　×　×　■  ■■■■■■■■  Queue: ゴールまで到達可能 |

## 問題２

|  |
| --- |
| 問題例ファイルを入力してください：maze1.txt  ■■■■■■  ■　×　　■  ■　　　×■  ■×　××■  ■　×　　■  ■■■■■■  Queue: ゴールまで到達不可能 |

|  |
| --- |
| 問題例ファイルを入力してください：maze2.txt  ■■■■■■■  ■　　××　■  ■　×　　　■  ■　　　×　■  ■××　×　■  ■　　　×　■  ■■■■■■■  Queue: ゴールまで到達可能  Path : (0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1, 2) (1, 3) (1, 4) (2, 4) (3, 4) (4, 4)  ■■■■■■■  ■●　××　■  ■●×●●●■  ■●●●×●■  ■××　×●■  ■　　　×●■  ■■■■■■■ |

|  |
| --- |
| 問題例ファイルを入力してください：maze3.txt  ■■■■■■■■  ■　×　　　　■  ■　　　×　×■  ■×　××　×■  ■　　　　　　■  ■×　×　×　■  ■　　×　×　■  ■■■■■■■■  Queue: ゴールまで到達可能  Path : (0, 0) (1, 0) (1, 1) (2, 1) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (4, 5) (5, 5)  ■■■■■■■■  ■●×　　　　■  ■●●　×　×■  ■×●××　×■  ■　●●●●●■  ■×　×　×●■  ■　　×　×●■  ■■■■■■■■ |

## 問題３

|  |
| --- |
| 問題例ファイルを入力してください：maze1.txt  ■■■■■■  ■　×　　■  ■　　　×■  ■×　××■  ■　×　　■  ■■■■■■  Queue: ゴールまで到達不可能  Stack: ゴールまで到達不可能 |

|  |
| --- |
| 問題例ファイルを入力してください：maze2.txt  ■■■■■■■  ■　　××　■  ■　×　　　■  ■　　　×　■  ■××　×　■  ■　　　×　■  ■■■■■■■  Queue: ゴールまで到達可能  Path : (0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1, 2) (1, 3) (1, 4) (2, 4) (3, 4) (4, 4)  ■■■■■■■  ■●　××　■  ■●×●●●■  ■●●●×●■  ■××　×●■  ■　　　×●■  ■■■■■■■Stack: ゴールまで到達可能  Path : (0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1, 2) (1, 3) (1, 4) (2, 4) (3, 4) (4, 4)  ■■■■■■■  ■●　××　■  ■●×●●●■  ■●●●×●■  ■××　×●■  ■　　　×●■  ■■■■■■■ |

|  |
| --- |
| 問題例ファイルを入力してください：maze3.txt  ■■■■■■■■  ■　×　　　　■  ■　　　×　×■  ■×　××　×■  ■　　　　　　■  ■×　×　×　■  ■　　×　×　■  ■■■■■■■■  Queue: ゴールまで到達可能  Path : (0, 0) (1, 0) (1, 1) (2, 1) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (4, 5) (5, 5)  ■■■■■■■■  ■●×　　　　■  ■●●　×　×■  ■×●××　×■  ■　●●●●●■  ■×　×　×●■  ■　　×　×●■  ■■■■■■■■Stack: ゴールまで到達可能  Path : (0, 0) (1, 0) (1, 1) (1, 2) (0, 2) (0, 3) (0, 4) (1, 4) (2, 4) (3, 4) (3, 5) (4, 5) (5, 5)  ■■■■■■■■  ■●×●●●　■  ■●●●×●×■  ■×　××●×■  ■　　　　●●■  ■×　×　×●■  ■　　×　×●■  ■■■■■■■■ |

# 考察

## デバックの効率化

　作成