|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *11* |  |
|  |  |  |  |  |  |  |  | *10* |  |
|  |  |  |  |  |  |  |  | *9* |  |
|  |  |  |  |  |  |  |  | *8* |  |
|  |  |  |  |  |  |  |  | *7* |  |
|  |  |  |  |  |  |  |  | *6* |  |
|  |  |  | *0* | *1* | *2* | *3* | *4* | *5* |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

int N[10][10]; //размер нашей карты

struct trajectory

{

int X;

int Y;

};

typedef struct trajectory Path;

Path root[50]; //назначаем массив с координамтами точек от первой до последней по которым надо идти, чтобы пройти маршрут по траектории

int map[][10] = {

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },

{ 0, 1, 1, 0, 0, 0, 1, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 0, 1, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 0, 0, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 1, 0, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },

{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }

};//карта местности, где "0" - пройти нельзя, "1" - пройти можно

map[7][3] = -1; //начало маршрута

map[3][6] = -2; //конец маршрута

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

N[i][j] = map[i][j];

if (map[i][j] == 0)

N[i][j] = 100; //назначаем непроходимым клеткам значение 100

if (map[i][j] == 1)

N[i][j] = 150; //назначем проходимым клеткам значение 150

if (map[i][j] == -2)

N[i][j] = 150; //назначаем концу пути 150

if (map[i][j] == -1)

N[i][j] = 0; //назначаем началу пути 0

}

}

for (int iter = 0; iter < 100; iter++) //строим волну, начиная из начала пути

{

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

if (N[i][j] == iter)

{

if ((i + 1)<10 && N[i + 1][j] != 100 && N[i + 1][j] > iter + 1)

N[i + 1][j] = iter + 1;

if ((i - 1) >= 0 && N[i - 1][j] != 100 && N[i - 1][j] > iter + 1)

N[i - 1][j] = iter + 1;

if ((j + 1)<10 && N[i][j + 1] != 100 && N[i][j + 1] > iter + 1)

N[i][j + 1] = iter + 1;

if ((j - 1) >= 0 && N[i][j - 1] != 100 && N[i][j - 1] > iter + 1)

N[i][j - 1] = iter + 1;

}

}

}

}

int znch0 = N[3][6]; //определяем колиество точек в маршруте

root[N[3][6]].X = 3; //координаты конца маршрута по Х

root[N[3][6]].Y = 6; //координаты конца маршрута по У

N[3][6] = -1; //задаем концу маршрута значение -1

for (int znch = (znch0 - 1); znch >= 0;) //ищем путь из конца в начало

{

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

if (N[i][j] == znch) //как только находим значение, на 1 меньшее чем предыдущее, тогда проверяем..

{

if (N[i + 1][j] == -1 || N[i - 1][j] == -1 || N[i][j + 1] == -1 | N[i][j - 1] == -1)//..чтобы клетка с этим значением была рядом по расположению с предыдущей клеткой..

{

root[N[i][j]].X = i; //..если это совпадает, и следующая клетка рядом с предыдщей и ее значение меньше на единицу, то записываем координаты этой новой точки

root[N[i][j]].Y = j;

N[i][j] = -1; //задаем клетке значение -1 и переходим к следующей точке и так, пока не придем в начало маршрута

znch = znch - 1;

}

}

}

}

}

int orient[2]; //массив с ориентацией робота в начальной точке

int coord[2]; //массив с координатами точки

int movements[100]; //массив со всеми движениями

orient[0] = -1; //ориентация робота..

orient[1] = 0; //..на свевер

int g,i;

for (i = 0, g = 0; i < znch0;) //определяем, какие команды подаем роботу (налево(1), прямо(2) или направо(3))

{

coord[0] = root[i + 1].X - root[i].X;

coord[1] = root[i + 1].Y - root[i].Y;

if (coord[0] == orient[0] && coord[1] == orient[1])

{

movements[g] = 2; //2 - идем прямо

i++;

g++;

}

else if (coord[1] == orient[0] && orient[1] == 0)

{

movements[g] = 1; //1 - поворачиваем налево при этих условиях и возвращаемся в начало алгоритма, чтобы пойти прямо

orient[0] = 0;

orient[1] = coord[1];

g++;

}

else if (coord[0] == orient[1] && orient[0] == 0)

{

movements[g] = 3; //3 - поворачиваем направо при этих условиях и возвращаемся в начало алгоритма, чтобы пойти прямо

orient[0] = coord[0];

orient[1] = 0;

g++;

}

else if (coord[1] == -orient[0] && orient[1] == 0)

{

movements[g] = 3; //3 - поворачиваем направо при этих условиях и возвращаемся в начало алгоритма, чтобы пойти прямо

orient[0] = 0;

orient[1] = coord[1];

g++;

}

else if (coord[0] == -orient[1] && orient[0] == 0)

{

movements[g] = 1; //1 - поворачиваем налево при этих условиях и возвращаемся в начало алгоритма, чтобы пойти прямо

orient[0] = coord[0];

orient[1] = 0;

g++;

}

}

for (int i = 0; i < g; i++)

{

printf("%2d ", movements[i]);

std::cout << std::endl;

} //выводим попорядку команды для робота, где 1 - поворот налево, 2 - идем прямо, 3 - поворот направо

std::cin.get();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  Эксперимента | 0 поворотов | 1 поворот | 2 поворота | 3 поворота |
| Результаты эксперимента | | |  |
| 1 | ✓ |  |  |  |
| 2 | ✓ |  |  |  |
| 3 |  | ✓ |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| Сумма правильных отработок |  |  |  |  |
| Сумма неправильных отработок |  |  |  |  |

Условные обозначения:

V – робот правильно отработал траекторию

Х – робот неправильно отработал траекторию