**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 18**

**«Лабиринт»**

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Основы алгоритмов и программирование»

|  |  |
| --- | --- |
| Преподаватель:  Яковлева Н.М.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил:  студент группы Y2231  Петров Н.В. |

Санкт-Петербург

2019/2020

**Цель работы:**

Составить программу, отыскивающую проход по лабиринту.

**Внутренние структуры данных:**

Текущее местоположение

struct Location {  
 short int x;  
 short int y;  
};



“Паук”, который бежит по лабиринту

struct Spider {  
 short int direction; // 1 - вниз, 2 - влево, 3 - вверх, 4 - вправо  
 Location location;  
};

Элемент стека

struct Node {  
 Location location;  
 Node \*p;  
};

**Алгоритм решения:**

С точки старта (x: 1, y: 0) «паук» начинает движение. В независимости от положения «паука» берется 3 величины с карты:

forward – клетка перед «пауком»

left/right – клетка справа/слева от паука

«Паук» в одну из свободных сторон согласно следующему приоритету: право, вперед, лево. В случае всех занятых клеток – он разворачивается. Получается что паук «идет» по правой стороне лабиринта. В зависимости от выбранного пути – направление паука меняется.

**Код:**

main.cpp

#include <iostream>  
#include "stack.h"  
  
using namespace std;  
  
#define DOWN 1  
#define LEFT 2  
#define UP 3  
#define RIGHT 4  
#define MAZE\_X\_SIZE 13  
#define MAZE\_Y\_SIZE 12  
  
int main() {  
  
 short int maze[MAZE\_Y\_SIZE][MAZE\_X\_SIZE] =  
 {{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
 {0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0},  
 {0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0},  
 {0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0},  
 {0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},  
 {0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0},  
 {0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0},  
 {0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0},  
 {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0},  
 {0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0},  
 {0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0},  
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0}};  
  
  
 struct Spider {  
 short int direction; // 1 - вниз, 2 - влево, 3 - вверх, 4 - вправо  
 Location location;  
 };  
  
 Spider spider = {DOWN, {1, 0}};  
  
 Node \*top = nullptr;  
 push(&top, spider.location);  
 cout << "y: " << spider.location.y << " x: " << spider.location.x << endl;  
  
 while (spider.location.y != MAZE\_Y\_SIZE - 1) {  
 bool forwd, left, right;  
 switch (spider.direction) {  
 case DOWN:  
 forwd = maze[spider.location.y + 1][spider.location.x];  
 left = maze[spider.location.y][spider.location.x + 1];  
 right = maze[spider.location.y][spider.location.x - 1];  
 break;  
 case LEFT:  
 forwd = maze[spider.location.y][spider.location.x - 1];  
 left = maze[spider.location.y + 1][spider.location.x];  
 right = maze[spider.location.y - 1][spider.location.x];  
 break;  
 case UP:  
 forwd = maze[spider.location.y + 1][spider.location.x];  
 left = maze[spider.location.y][spider.location.x - 1];  
 right = maze[spider.location.y][spider.location.x + 1];  
 break;  
 case RIGHT:  
 forwd = maze[spider.location.y][spider.location.x + 1];  
 left = maze[spider.location.y - 1][spider.location.x];  
 right = maze[spider.location.y + 1][spider.location.x];  
 break;  
 }  
 if (right) {  
 switch (spider.direction) {  
 case UP:  
 spider.direction = RIGHT;  
 spider.location.x++;  
 break;  
 case DOWN:  
 spider.direction = LEFT;  
 spider.location.x--;  
 break;  
 case LEFT:  
 spider.direction = UP;  
 spider.location.y--;  
 break;  
 case RIGHT:  
 spider.direction = DOWN;  
 spider.location.y++;  
 break;  
 }  
 } else if (forwd) {  
 switch (spider.direction) {  
 case UP:  
 spider.location.y--;  
 break;  
 case DOWN:  
 spider.location.y++;  
 break;  
 case LEFT:  
 spider.location.x--;  
 break;  
 case RIGHT:  
 spider.location.x++;  
 break;  
 }  
 } else if (left) {  
 switch (spider.direction) {  
 case UP:  
 spider.direction = LEFT;  
 spider.location.x--;  
 break;  
 case DOWN:  
 spider.direction = RIGHT;  
 spider.location.x++;  
 break;  
 case LEFT:  
 spider.direction = DOWN;  
 spider.location.y++;  
 break;  
 case RIGHT:  
 spider.direction = UP;  
 spider.location.y--;  
 break;  
 }  
 } else {  
 spider.direction -= 2;  
 if (spider.direction == 0) spider.direction = 4;  
 if (spider.direction == -1) spider.direction = 3;  
 }  
 push(&top, spider.location);  
  
 cout << "y: " << spider.location.y << " x: " << spider.location.x << " direction: ";  
 switch (spider.direction) {  
 case UP:  
 cout << "up";  
 break;  
 case DOWN:  
 cout << "down";  
 break;  
 case LEFT:  
 cout << "left";  
 break;  
 case RIGHT:  
 cout << "right";  
 break;  
 }  
 cout << endl;  
 if (!maze[spider.location.y][spider.location.x]) {  
 return 1;  
 }  
 }  
 cout << "Выход есть!" << endl;  
  
 while (top) {  
 Location val = pop(&top);  
 maze[val.y][val.x] = 2;  
 }  
 for (auto &y : maze) {  
 for (int x : y) {  
 switch (x) {  
 case 0:  
 cout << 'N';  
 break;  
 case 1:  
 cout << ' ';  
 break;  
 default:  
 cout << '+';  
 break;  
 }  
 }  
 cout << endl;  
 }  
 return 0;  
}

stack.h

#ifndef LAB18\_STACK\_H  
#define LAB18\_STACK\_H  
  
struct Location {  
 short int x;  
 short int y;  
};  
  
struct Node {  
 Location location;  
 Node \*p;  
};  
  
void push(Node \*\*top, Location location);  
Location pop(Node \*\*top);  
  
#endif //LAB18\_STACK\_H

stack.cpp

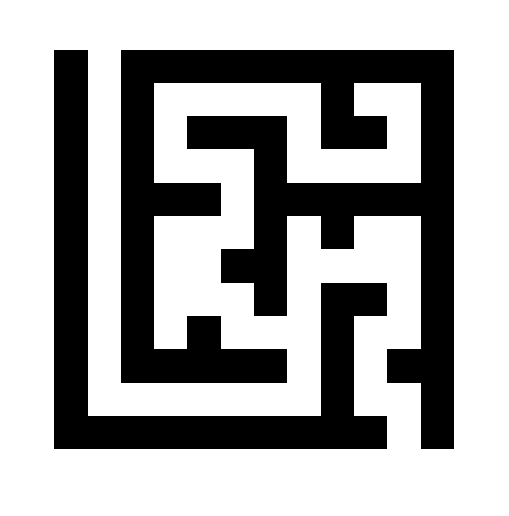
#include "stack.h"  
  
void push(Node \*\*top, const Location location)  
{  
 Node \*pv = new Node;  
 pv->location = location;  
 pv->p = \*top;  
 \*top = pv;  
}  
  
Location pop(Node \*\*top)  
{  
 Location temp = (\*top)->location;  
 Node \*pv = \*top;  
 \*top = (\*top)->p;  
 delete pv;  
 return temp;  
}

**Результаты и вывод:**

Вывод программы

y: 0 x: 1  
y: 1 x: 1 direction: down  
y: 2 x: 1 direction: down  
y: 3 x: 1 direction: down  
y: 4 x: 1 direction: down  
y: 5 x: 1 direction: down  
y: 6 x: 1 direction: down  
y: 7 x: 1 direction: down  
y: 8 x: 1 direction: down  
y: 9 x: 1 direction: down  
y: 10 x: 1 direction: down  
y: 10 x: 2 direction: right  
y: 10 x: 3 direction: right  
y: 10 x: 4 direction: right  
y: 10 x: 5 direction: right  
y: 10 x: 6 direction: right  
y: 10 x: 7 direction: right  
y: 9 x: 7 direction: up  
y: 8 x: 7 direction: up  
y: 7 x: 7 direction: up  
y: 6 x: 7 direction: up  
y: 6 x: 8 direction: right  
y: 6 x: 9 direction: right  
y: 6 x: 10 direction: right  
y: 7 x: 10 direction: down  
y: 8 x: 10 direction: down  
y: 8 x: 9 direction: left  
y: 9 x: 9 direction: down  
y: 10 x: 9 direction: down  
y: 10 x: 10 direction: right  
y: 11 x: 10 direction: down  
Выход есть!  
N+NNNNNNNNNNN  
N+N N NN  
N+N NNN NN NN  
N+N N NN  
N+NNN NNNNNNN  
N+N N N NN  
N+N NN++++NN  
N+N N+NN+NN  
N+N N +N++NN  
N+NNNNN+N+NNN  
N+++++++N++NN  
NNNNNNNNNN+NN

Программа находит выход в введённом лабиринте.



(Введённый лабиринт)