**南昌航空大学**

**20学年—21学年第 2 学期 智能医疗信息处理专业实验**

**实验一**

专业名称： 生物医学工程 实验学时： 2

学号： 19084127 姓名： 周亚诺

实验题目：栈的应用－区域填充

实验环境： CLion

实验目的：

1．掌握栈的定义及实现；

2．掌握利用栈实现区域填充。

实验内容：

（1）通过二维数组输入一幅边界图像；

（2）建立图像坐标数据结构；

（3）建立图像像素点栈；

（4）从点击位置出发，通过进栈和出栈操作实现边界内或外的区域填充。

实验要求：

(1) 程序要添加适当的注释，程序的书写要采用缩进格式。

(2) 程序要具在一定的健壮性，即当输入数据非法时，程序也能适当地做出反应。

(3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。

(4) 根据实验报告模板详细书写实验报告,在实验报告中给出表达式求值算法的流程图。

实验代码：

1. MyStack.h

#ifndef CPP\_STACKFILL\_MYSTACK\_H  
#define CPP\_STACKFILL\_MYSTACK\_H  
  
#include <iostream>  
#include <windows.h>  
  
using namespace std;  
  
template<class T>class MyStack {  
public:  
 struct Node {  
 T data;  
 Node\* next = nullptr;  
 };  
  
 Node \*head;  
 Node \*topNode;  
 int length;  
  
 MyStack() {  
 head = NULL;  
 length = 0;  
 }  
  
 void push(T n) {  
 Node\* temp = new Node;  
 temp->data = n;  
  
 if (head == NULL) {  
 temp->next = head;  
 head = temp;  
 topNode = temp;  
 } else {  
 temp->next = topNode;  
 topNode = temp;  
 }  
  
 length++;  
 }  
  
 T pop() {  
 if (length <= 0) {  
 abort();  
 }  
  
 Node\* q = topNode;  
 T data = topNode->data;  
 topNode = topNode->next;  
 delete(q);  
 length--;  
  
 return data;  
 }  
  
 int size() {  
 return length;  
 }  
  
 T top() {  
 return topNode->data;  
 }  
  
 bool isEmpty() {  
 if (length == 0) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 void clear() {  
 while (length) {  
 pop();  
 }  
 }  
};  
  
  
#endif //CPP\_STACKFILL\_MYSTACK\_H

1. MyPoint.h

#ifndef CPP\_STACKFILL\_MYPOINT\_H  
#define CPP\_STACKFILL\_MYPOINT\_H  
  
#include "MyStack.h"  
  
class MyPoint {  
private:  
 int row;  
 int col;  
  
public:  
 MyPoint(int row, int col) : row(row), col(col) {}  
 MyPoint() = default;  
  
 int getRow() const {  
 return row;  
 }  
  
 int getCol() const {  
 return col;  
 }  
};  
  
  
#endif //CPP\_STACKFILL\_MYPOINT\_H

1. MyMaze.h

#ifndef CPP\_STACKFILL\_MYMAZE\_H  
#define CPP\_STACKFILL\_MYMAZE\_H  
  
#define MAZE\_SIZE 10  
  
#include "MyPoint.h"  
  
class MyMaze {  
public:  
 int maze[MAZE\_SIZE][MAZE\_SIZE] = {  
 {2, 2, 2, 2, 2, 2, 2, 2, 2, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 0, 0, 0, 0, 0, 0, 0, 0, 2},  
 {2, 2, 2, 2, 2, 2, 2, 2, 2, 2}  
 };  
 MyStack<MyPoint> myStack;  
 MyPoint startPoint;  
  
 void printMaze();  
 void fill();  
 bool isFull();  
 MyMaze(const MyPoint &startPoint);  
};  
  
  
#endif //CPP\_STACKFILL\_MYMAZE\_H

1. MyMaze.cpp

#include "MyMaze.h"  
  
void MyMaze::printMaze() {  
 for (int i = 0; i < MAZE\_SIZE; i++) {  
 for (int j = 0; j < MAZE\_SIZE; j++) {  
 cout << maze[i][j] << " ";  
 cout.setf(ios\_base::unitbuf);  
 }  
 cout << endl;  
 }  
}  
  
void MyMaze::fill() {  
 myStack.push(startPoint);  
  
 while (!myStack.isEmpty()) {  
 MyPoint curLocation = myStack.top();  
 const int curRow = curLocation.getRow();  
 const int curCol = curLocation.getCol();  
 maze[curRow][curCol] = 3;  
  
 if (maze[curRow-1][curCol] == 0) {  
 // if point can move north  
 myStack.push(MyPoint(curRow-1, curCol));  
 maze[curRow][curCol] = 1;  
 } else if (maze[curRow][curCol-1] == 0) {  
 // if point can move west  
 myStack.push(MyPoint(curRow, curCol-1));  
 maze[curRow][curCol] = 1;  
 } else if (maze[curRow+1][curCol] == 0) {  
 // if point can move south  
 myStack.push(MyPoint(curRow+1, curCol));  
 maze[curRow][curCol] = 1;  
 } else if (maze[curRow][curCol+1] == 0) {  
 // if point can move east  
 myStack.push(MyPoint(curRow, curCol+1));  
 maze[curRow][curCol] = 1;  
 } else {  
 myStack.pop();  
 }  
  
 cout << "moving..." << endl;  
 Sleep(500);  
 printMaze();  
  
 if (isFull()){  
 break;  
 }  
 }  
}  
  
MyMaze::MyMaze(const MyPoint &startPoint) : startPoint(startPoint) {}  
  
bool MyMaze::isFull() {  
 for (int i = 0; i < MAZE\_SIZE; i++) {  
 for (int j = 0; j < MAZE\_SIZE; j++) {  
 if (maze[i][j] == 0) {  
 return false;  
 }  
 }  
 }  
 return true;  
}

1. main.cpp

#include "MyMaze.h"  
  
int main() {  
 MyMaze myMaze(MyPoint(3, 3));  
 myMaze.printMaze();  
 myMaze.fill();  
  
 return 0;  
}

心得体会：

使用模板类可以让Stack可以复用，涉及类似的结构可以减少代码量。