数据结构课程设计

项目说明文档

勇闯迷宫游戏

软件工程

张靖凯

2151396

目录

项目要求

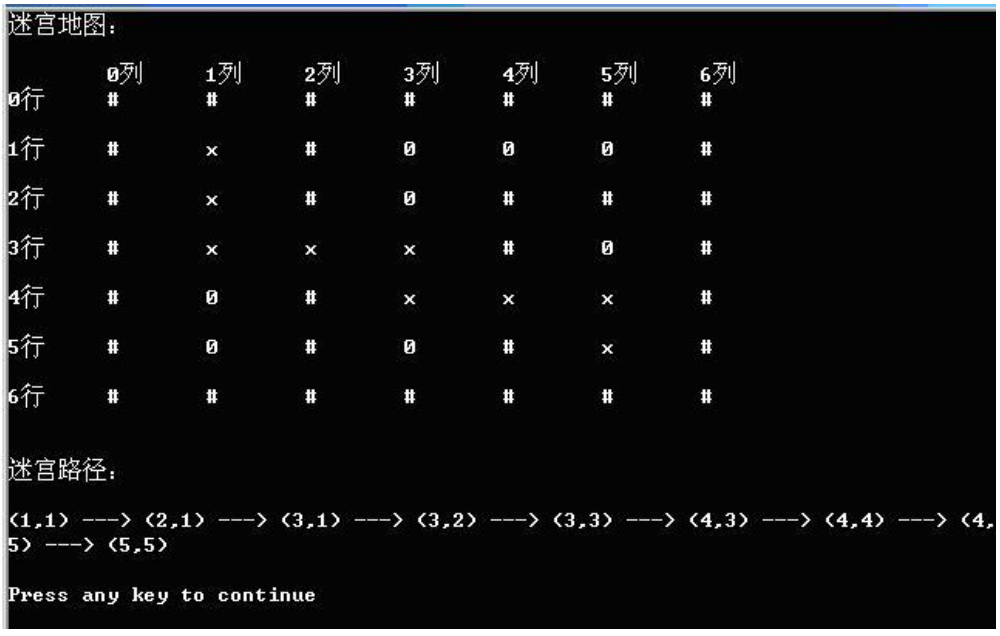
项目简介

迷宫只有两个门，一个门叫入口，另一个门叫出口。一个骑士骑马从入口进入迷宫，迷宫设置很多障碍，骑士需要在迷宫中寻找通路以到达出口。

项目功能要求

迷宫问题的求解过程可以采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。

项目示例



项目设计

数据结构与类设计

设计了配有迭代器的 vector 容器。

由于 vector 在双向移动的基础上，支持前后位置的比较、随机存取、直接移动 n 个距离，故迭代器类型为 random-access iterator，为了方便，直接 using iterator = T*来实现，因为指针有如上的性质。

实现了默认构造、拷贝构造、拷贝赋值。重载了[]运算符，操作类函数模拟 STL 实现。

Detailed Description

template<typename T>
class MercedesKK::Vector< T >

自行实现的vector容器

实现了元素数量超出_size时容量翻倍的操作 简化，迭代器在Vector类内直接实现

Public Types

using	**value_type** = T
using	**iterator** = T *

Public Member Functions

iterator	**begin** ()
iterator	**end** ()
Constructors	
	Vector ()
	~Vector ()
	Vector (const **Vector** &vec)
Vector &	**operator=** (const **Vector** &vec)
重载运算符	
value_type &	**operator[]** (size_t index)
bool	**operator==** (const **Vector** &vec) const
Operations	
void	**push_back** (**value_type** val)
void	**pop_back** ()
size_t	**Size** () const
size_t	**capacity** () const
bool	**empty** ()
void	**clear** ()
value_type	**front** () const
value_type	**back** () const
void	**insert** (**iterator** it, **value_type** val)
void	**erase** (**iterator** it)

Private Attributes

value_type *	**_data** 动态分配实现的数组 More...
size_t	**_size** 已有元素数量 More...
size_t	**_capacity** 容器容量 More...

解题思路及迷宫 DFS 算法

```
char maze[10][10] =
{
    "#######",
    "#0#000#",
    "#0#0###",
    "#00000#",
    "#0###0#",
    "#0#0#0#",
    "#######",
};
```

采用深搜+回溯完成递归搜索出口。

本题的地图采用写死的方式，故采用二维数组来实现地图。但是入口出口可以自己输入。

使用 Vector 来存储每一个经过的坐标，实现输出。

在 dfs 函数中，递归终止条件为找到出口，将 tag 设置为 1；若没有找到出口则像四个方向遍历递归，调用递归函数之前要先将路线存入 vector 中（防止逆序输出），在本层递归搜索结束后，如果 tag 值为 1（代表找到出口）则直接 return，最后进行回溯，即 vector 调用 erase 函数抹去此次路线。

函数实现如下：

```
void dfs(int x, int y)
{
    if (x == exitPos.x && y == exitPos.y)
    {
        tag = 1;
        maze[x][y] = 'x';
        route.push_back(exitPos);
        return;
    }

    for (int i = 0; i < 4; i++)
    {
        if (maze[x + direction[i].x][y + direction[i].y] == '0')
        {
            maze[x][y] = 'x';
            Position cur = { x,y };
            route.push_back(cur);

            dfs(x + direction[i].x, y + direction[i].y);

            if (tag == 1)
                return;

            // 回溯
            route.erase(route.end() - 1);
            maze[x][y] = '0';
        }
    }
}
```

设计亮点

代码注释规范

采用了 doxygen 注释规范，对类、函数等有简要说明，命名采用驼峰命名法，类内的各个声明规范，方便本人回顾之前写过的代码和 code reviewer 查看，使 API 规范，帮助这个开发流程高效、规范地进行。

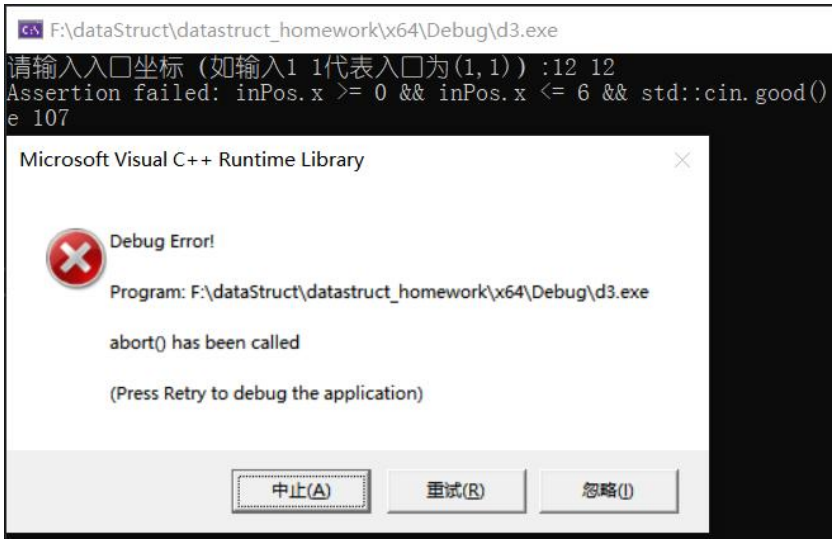
```
/// @name Constructors
/// @{
Vector() :_data(nullptr), _size(0), _capacity(0) {}
~Vector();
Vector(const Vector& vec);
Vector& operator=(const Vector& vec);
/// @}
// end of Constructors

/// @name 重载运算符
/// @{
value_type& operator[](size_t index) { return _data[index]; };
bool operator==(const Vector& vec) const;
/// @}
// end of overloads


/// @name Operations
/// @{
///
```

输入错误处理

```
assert(inPos.x >= 0 && inPos.x <= 6 && std::cin.good())
```



项目测试

```
F:\dataStruct\datastruct_homework\x64\Debug\d3.exe
请输入入口坐标（如输入1 1代表入口为(1,1)）:1 1
请输入出口坐标（如输入5 5代表入口为(5,5)）:5 5
迷宫地图:
        0列     1列     2列     3列     4列     5列     6列
0行     #       #       #       #       #       #       #

1行     #       x       #       0       0       0       #

2行     #       x       #       0       #       #       #

3行     #       x       x       x       x       x       #

4行     #       0       #       #       #       x       #

5行     #       0       #       0       #       x       #

6行     #       #       #       #       #       #       #

迷宫路径:
<1,1> ---> <2,1> ---> <3,1> ---> <3,2> ---> <3,3> ---> <3,4> ---> <3,5> ---> <4,5> ---> <5,5>

Enter to Exit
```

Linux 下测试（Ubuntu）

```
kk@LAPTOP-UJDPHKT8: ~/da
  1 #include <iostream>
  2 #include <string>
  3 #include <cassert>
  4 #include "Vector.hpp"
  5
  6 using namespace MercedesKK;>///< use my namespace
  7
  8 struct Position
  9 {
 10     int x;>-///< 坐标x
 11     int y;>-///< 坐标y
 12 };
 13
 14 /// 为解题方便 开全局变量
 15 /// maze 是从左上角 (0,0) 开始的!
 16 char maze[10][10] =-
 17 {
 18     "#######",
 19     "#0#000#",
 20     "#0#0###",
 21     "#00000#",
 22     "#0###0#",
 23     "#0#0#0#",
 24     "#######",
 25 };-
 26 const Position direction[4] = { {0,1}, {0,-1} ,{1,0},{-1,0} };
 27 /// 出口位置
h3.cpp
"h3.cpp" 143L, 3332C
```

```
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework2$ cd ..
kk@LAPTOP-UJDPHKT8:~/dataStruct$ cd homework3
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework3$ ls
Vector.hpp  h3.cpp  run
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework3$ vim h3.cpp
kk@LAPTOP-UJDPHKT8:~/dataStruct/homework3$ ./run
请输入入口坐标（如输入1 1代表入口为(1,1)）:1 1
请输入出口坐标（如输入5 5代表入口为(5,5)）:5 5
迷宫地图：
        0列     1列     2列     3列     4列     5列     6列
0行     #       #       #       #       #       #       #

1行     #       x       #       0       0       0       #

2行     #       x       #       0       #       #       #

3行     #       x       x       x       x       x       #

4行     #       0       #       #       #       x       #

5行     #       0       #       0       #       x       #

6行     #       #       #       #       #       #       #

迷宫路径：
<1,1> ---> <2,1> ---> <3,1> ---> <3,2> ---> <3,3> ---> <3,4> ---> <3,5> ---> <4,5> ---> <5,5>

Enter to Exit
```