# 勇闯迷宫游戏

## 1.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。

## 2.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。

## 3.核心功能及代码

- **声明的数据结构**

```
char map[7][7]={                        //内置地图
        '#','#','#','*','#','#','#',
        '#','*','#','*','*','*','#',
        '*','*','#','#','#','#','#',
        '#','*','*','*','#','*','#',
        '*','*','#','*','*','*','#',
        '#','*','#','*','#','*','#',
        '#','#','#','#','#','#','#'
};
int xx[4]={1,0,-1,0};                   //定义四个方向的bias，方便搜索时用for循环代替
int yy[4]={0,1,0,-1};                   //同上

class point{                            //定义搜索顶点的类
public:
    int x;
    int y;
};

vector<point>answer;                                    //保存结果搜寻路径
```

- **打印路径函数**
  由于路径是不断加入到answer中，最后把不正确的顶点都丢掉的，所以最后剩下的顶点就是路径，因此我们直接遍历整个数组将其打印即可。

```
void display(){                         //打印路线图
    int count=0;
    cout<<"迷宫路径:"<<endl;
    for(auto i:answer){
        cout<<'('<<i.x+1<<','<<i.y+1<<')';
        count++;
        if(count!=answer.size()){       //防止在最后一个顶点时打印出--->
            cout<<"--->";
        }
    }
}
```

- **深度优先搜索函数**
  最朴素的dfs函数，这里做了一点优化，防止重复搜索和冗余的数据结构
  - 通过在进行下一步dfs之前将加入的顶点设为不可走，在返回后设置为可走，就可以不用在加入时记录加入顶点后搜索的方向。
  - 通过在对四个方向搜索未找到可行路径后直接将该节点弹出的同时把该节点设置为不可走的，从而减少了后序搜索时不必要的搜素
  - 这里构造了两个数组，用biasX和biasY与当前顶点位置相加从而将4个不同的方向压缩到一个for循环中，简化了代码。

```
void dfs(){                                 //深度优先搜索函数
    auto Point=answer.back();               //获得路径的上一步顶点
    int PointX=Point.x;
    int PointY=Point.y;
    if(!((PointX==6&&PointY==6))){          //如果没有到达出口就继续寻找
        for(int i=0;i<4;i++){               //对四个加bias进行方向搜索
            int X=PointX+xx[i];
            int Y=PointY+yy[i];
            if(X>=0&&X<7&&Y>=0&&Y<7) {      //检测是否超出边界
                if (map[X][Y] == '#') {     //如果探测的此顶点可行就加入answer中
                    auto ptr = new point;
                    ptr->x = X;
                    ptr->y = Y;
                    answer.push_back(*ptr);
                    map[X][Y] = '*';        //把加入的顶点设为不可走防止后来重复搜索
                    dfs();
                    map[X][Y] = '#';        //如果返回后重新设为可走
                }
            }
        }
        map[PointX][PointY]='*';            //如果四个方向都走不通就返回并把该顶点设为不可走因为搜索过不通
        answer.pop_back();                  //从answer中取出该节点
        return;
    }
    else{
        display();                              //如果到达出口则打印路径
        exit(0);
    }
}
```

# 4.项目实例演示

```
/Users/kirito/CLionProjects/untitled/cmake-build-debug/untitled
输入1为自带地图，输入2为自定义地图
迷宫路径:
(1,1)--->(1,2)--->(1,3)--->(2,3)--->(3,3)--->(3,4)--->(3,5)--->(3,6)--->(3,7)--->(4,7)--->(5,7)--->(6,7
Process finished with exit code 0
```