Linux小实战2-五子棋下

> - 巩固Linux环境指令操作
> - 巩固Linux中相关开发工具的基本使用
> - 练习在Linux环境当中进行C编程
> - 编写五子棋程序下层基本游戏逻辑
> - 完成项目，扩展项目，提供扩展思路，总结项目

第五步. 实现游戏落子逻辑

```
void PlayerMove(int board[ROW][COL], int who)
{
    while(1){
        printf("\nPlayer[ %d ] Please Enter Your Postion# ", who);
        scanf("%d%d", &x, &y);
        if(x<0 || y > COL){    //这里的x，y是全局的，每次都记录着当前用户的落子位置
            printf("Postion Error!\n");
        }
        else if(board[x][y] != 0){
            printf("Postion Is Occupied!\n");
        }
        else{
            board[x][y] = who;    //谁落子，就放置谁的数据
            break;
        }
    }
}
```

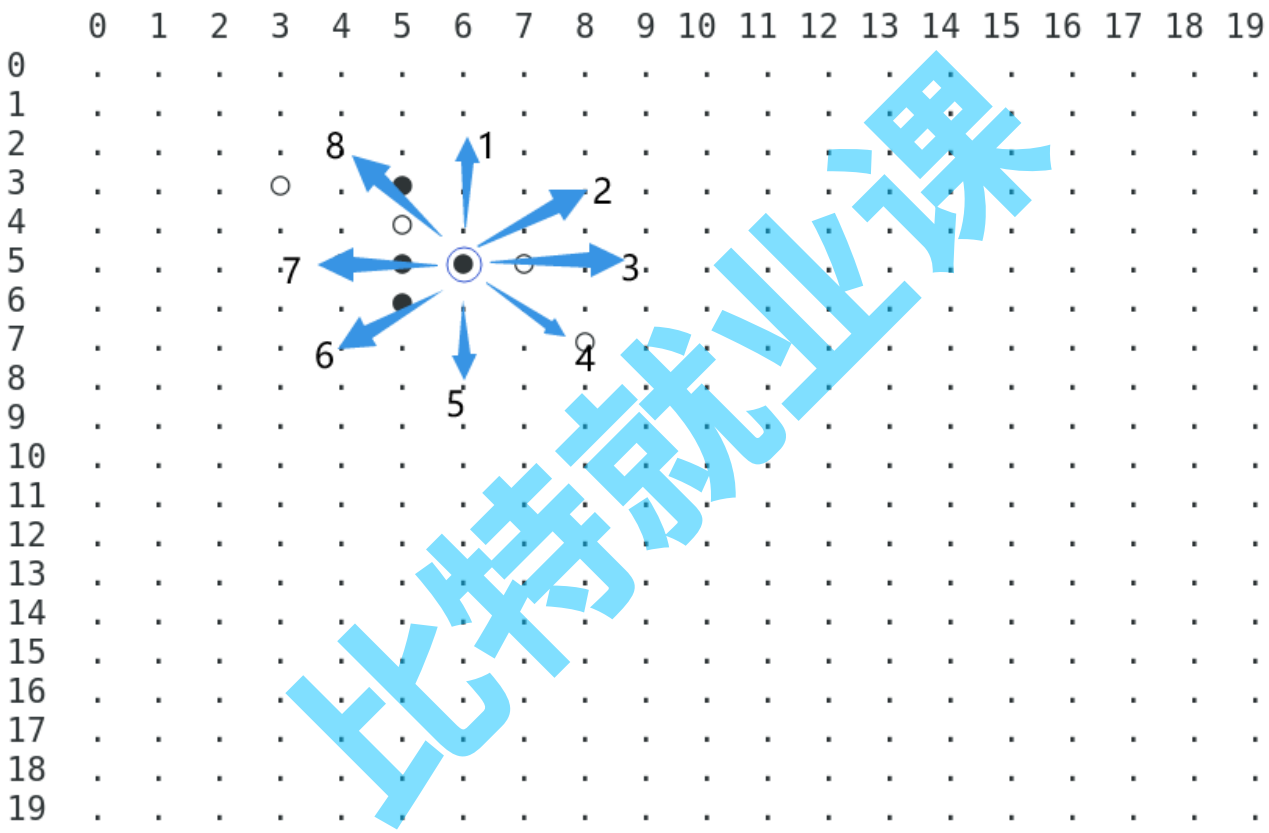
第六步. 实现游戏的显示逻辑

```
void ShowBoard(int board[ROW][COL])
{
    printf("\033c");
    printf("\n\n  ");
    for(int i = 0; i < COL; i++){
        printf("%3d", i);
    }
    printf("\n");
    for(int i = 0; i < ROW; i++){
        printf("%-3d", i);
        for(int j = 0; j < COL; j++){
            if(board[i][j] == PLAYER1){
               //player1
                printf(" • ");
            }
            else if(board[i][j] == PLAYER2){
                //player2
                printf(" ○ ");
            }
```

```
            else{
                //Space
                printf(" . ");
            }
        }
        printf("\n");
    }
}

//默认符号比较丑陋，我们需要使用美化后的符号,可以在下面的链接处获取符号，直接复制粘贴就行
//https://www.jiuwa.net/fuhao/daquan/
```

第七步. 判定五子连珠然后判定游戏结果



任何落子位置都有八个方向，所以判定五子连珠，本质是判定1,5方向之和，2,6方向之和，3，7方向之和，4,8方向之和，其中任意一个出现相同的连续五个棋子，即游戏结束

```
enum Dir{        // 枚举八个方向
    LEFT,
    RIGHT,
    UP,
    DOWN,
    LEFT_UP,
    RIGHT_DOWN,
    RIGHT_UP,
    LEFT_DOWN
};
```

```c
int ChessCount(int board[ROW][COL], enum Dir d)    //这个函数是重点， 需要给学生单独画图慢慢解释
{
    int _x = x;
    int _y = y;

    int count = 0;
    while(1){
      switch(d){
        case LEFT:
          _y--;
          break;
        case RIGHT:
          _y++;
          break;
        case UP:
          _x--;
          break;
        case DOWN:
          _x++;
          break;
        case LEFT_UP:
          _x--, _y--;
          break;
        case RIGHT_DOWN:
          _x++, _y++;
          break;
        case RIGHT_UP:
          _x--, _y++;
          break;
        case LEFT_DOWN:
          _x++, _y--;
          break;
      }
      if(_x < 0 || _x > ROW-1 || _y < 0 || _y > COL-1){ //坐标移动完毕，一定要先保证没有越界
          break;
      }
      if(board[x][y] == board[_x][_y]){ //判定指定位置和原始位置的棋子是否相同，“连珠”就体现在这里
          count++;
      }
      else{
          break;
      }
    }
    return count;
}
int IsOver(int board[ROW][COL])
{
    //见上图，注意，每次统计的时候，都没有统计当前节点，需要单独+1
    int count1 = ChessCount(board, LEFT) + ChessCount(board, RIGHT) + 1;
    int count2 = ChessCount(board, UP) + ChessCount(board, DOWN) + 1;
    int count3 = ChessCount(board, LEFT_UP) + ChessCount(board, RIGHT_DOWN) + 1;
    int count4 = ChessCount(board, LEFT_DOWN) + ChessCount(board, RIGHT_UP) + 1;
```

```
    if(count1 >= 5 || count2 >= 5 || count3 >= 5 || count4 >= 5){
      if(board[x][y] == PLAYER1){
          return PLAYER1_WIN;
      }
      else{
          return PLAYER2_WIN;
      }
    }
    for(int i = 0; i < ROW; i++){
        for(int j = 0; j < COL; j++){
            if(board[i][j] == 0){
                return NEXT;
            }
        }
    }
    return DRAW;
}
```

第八步. 联合调试

- 保证程序能够编写通过
- 设计测试用例

第九步. 扩展

- 引入进度条
- 尝试人机对战
- 功能扩展：颜色提示，步数记录，先手随机交换等
- 网络版本?

第十步. 完整代码链接

```
https://gitee.com/whb-helloworld/gobang
#github太卡了，我们使用码云
```