- void main(void);
  - 初始化绘图窗口
    - initgraph(960,488);
  - //显示主界面
    - void DispMenu(void);
      - //加载主界面
        - loadimage();
      - //显示主界面
        - putimage();
  - //返回用户用鼠标交互的选择（1开始游戏，2退出游戏，3游戏规则）
    - char ChooseMenu(void);
      - if 鼠标点左键
        - 点击位置在"开始游戏"按钮范围内 → 返回 '1'
        - 点击位置在"结束游戏"按钮范围内 → 返回 '2'
        - 点击位置在"游戏说明"按钮范围内 → 返回 '3'
  - void ChooseGame(void)
    - 选择游戏模式
      - if 鼠标点左键，点击位置在"人人对战"按钮范围内，Game(0)
      - if 鼠标点左键，点击位置在"人机对战"按钮范围内，Game(1)
    - void Game(int i);
      - //1、游戏竞技
        - typedef struct chess
          ```
          {
              int num;//棋子序号，也是棋子的正面的图片文件名
              int point;//棋子角色的大小，如兵是0，马是2，马可吃兵，兵不能吃马
              char color;//棋子角色的颜色
          }CHESS;
          ```
        - typedef struct click
          ```
          {
              int num;//棋子序号，也是棋子的正面的图片文件名
              int point;//棋子角色的大小，如兵是0，马是2，马可吃兵，兵不能吃马
              char color;//棋子角色的颜色
              int map;//地图数组下标对应的棋盘图元素序号，也是棋盘图元素的图片文件名（它的值是不变的，因为棋盘图元素的布局是确定不变的）
              bool overturn;//是否已经被翻转过
          }MAP;
          ```
        - typedef struct choosed
          ```
          {
              int i;//行坐标
              int j;//列坐标
          }CHOOSED;
          ```
        - 全局结构体，用来通过已知序号查询point和color的值
          - CHESS Library[32]={{0,0,'B'},{1,0,'B'},{2,0,'B'},{3,0,'B'},{4,0,'B'},{5,1,'B'},{6,1,'B'},{7,2,'B'},
            {8,2,'B'},{9,3,'B'},{10,3,'B'},{11,4,'B'},{12,4,'B'},{13,5,'B'},{14,5,'B'},{15,6,'B'},
            {16,0,'R'},{17,0,'R'},{18,0,'R'},{19,0,'R'},{20,0,'R'},{21,1,'R'},{22,1,'R'},{23,2,'R'},
            {24,2,'R'},{25,3,'R'},{26,3,'R'},{27,4,'R'},{28,4,'R'},{29,5,'R'},{30,5,'R'},{31,6,'R'});
        - 全局变量，记录棋盘剩余棋子数
          - int numofB=16;//棋盘剩余黑色棋子总数，初始为16
            int numofR=16;//棋盘剩余红色棋子总数，初始为16
        - 全局变量，存Game中所需图片的地址
          - IMAGE img[9];//棋盘图元素组（9张）
            imgback;//游戏界面

            //使用三元光栅操作实现透明贴图
            //掩码图：与原图对应，原图需要透明的部分，用纯黑色表示；原图需要显示的部分，用纯白色表示。
            chessblack;//图（黑底白图）
            //原图：需要透明的部分，用纯白色表示。
            chessimg[32];//棋子正面图片组（32张）
            chessback;//棋子背面（白底彩图）

            youwin;//你赢了的界面
            youlose;//你输了的界面
            player1win;//玩家1赢了的界面
            player2win;//玩家2赢了的界面
            draw;//平局的界面

            chessboard;//游戏界面中右侧栏显示选中的棋子的区域的底图
            Player;//游戏界面中右侧栏显示"玩家"两字
            name1;//游戏界面中右侧栏显示"Player1"两字
            name2;//游戏界面中右侧栏显示"Player2"两字
        - Game()函数中的结构体，用来存储棋盘中各块位置的状态信息
          - //结构数组初始化 num=-1;point=-1;color='A';map的值由棋盘的布局决定;overturn=false;
            - MAP Location[32]={{-1,-1,'A',8,false},{-1,-1,'A',8,false},{-1,-1,'A',8,false},{-1,-1,'A',8,false},{-1,-1,'A',0,false},{-1,-1,'A',1,false},{-1,-1,'A',8,false},{-1,-1,'A',8,false},{-1,-1,'A',8,false},
            {-1,-1,'A',5,false},{-1,-1,'A',3,false},{-1,-1,'A',8,false},{-1,-1,'A',8,false},{-1,-1,'A',1,false},{-1,-1,'A',0,false},{-1,-1,'A',8,false},{-1,-1,'A',5,false},{-1,-1,'A',3,false},
            {-1,-1,'A',7,false},{-1,-1,'A',6,false},{-1,-1,'A',6,false},{-1,-1,'A',3,false},{-1,-1,'A',3,false},{-1,-1,'A',5,false},{-1,-1,'A',5,false},{-1,-1,'A',5,false},{-1,-1,'A',7,false},{-1,-1,'A',6,false},
            {-1,-1,'A',2,false},{-1,-1,'A',4,false},{-1,-1,'A',2,false},{-1,-1,'A',4,false},{-1,-1,'A',2,false},{-1,-1,'A',4,false},{-1,-1,'A',2,false},{-1,-1,'A',4,false});
        - //加载棋盘界面所有需要的图片
          - void Loadback(void); → loadimage();
        - //画初始棋盘
          - void Print(void); → putimage();
        - void RandChess(MAP Location[32]); → 生成32个不相同的随机数存入int num[32];
          - //随机洗子摆子
          - 随机数在0 1 ... 31中取
          - 用随机得到的序号排序，在Library里查询，并更新Location中的参数
            - Location[0~31].num=num[0~31];
              Location[0~31].point=Library[num[0~31]].point;
              Location[0~31].color=Library[num[0~31]].color;
        - void Go(MAP Location[32]);
          - //开始人人对战
          - 记录玩家的棋子对应的颜色信息
            - char Player1,Player2;
            - void DispChess(MAP Location[32],char *color);
              - 点击翻棋
            - void Rule(MAP Location[32],char color)
              - 玩家行棋
        - void AlphaGo(MAP Location[32]);
          - //开始人机对战
          - 记录玩家的棋子对应的颜色信息
            - char Player1,Player2;
            - void DispChess(MAP Location[32],char *color);
              - 点击翻棋
            - void AiRule(MAP Location[32],char color)
              - 电脑行棋
            - void Rule(MAP Location[32],char color)
              - 玩家行棋
        - 当出现输赢结果界面时，不跳出函数，按任意键后才跳出函数，即返回游戏菜单主界面
          - getch();
  - //2、关闭图形环境
    - void Closegraph(void);
      - //加载结束界面图片
        - loadimage();
      - //显示结束界面图片
        - putimage();
  - //3、游戏规则说明，载入并显示规则图片，按任意键返回
    - int Introduce(void);
      - //加载游戏规则图片
        - loadimage();
      - //显示游戏规则图片
        - putimage();
      - //显示规则后不跳出函数，用户按任意键时，跳出函数，即返回游戏菜单主界面
        - getch();