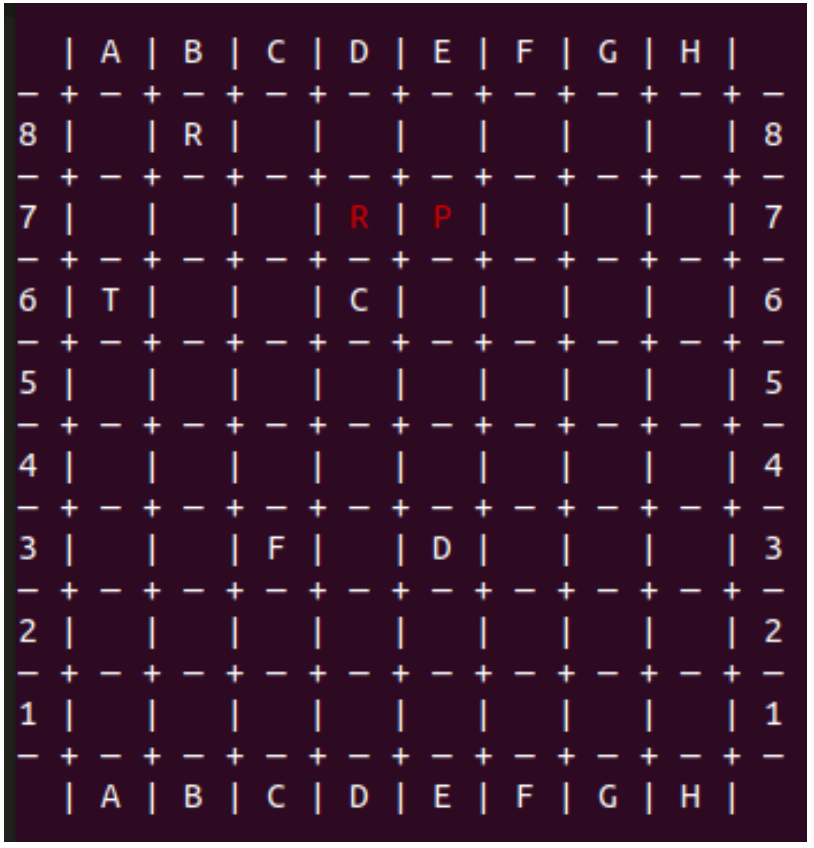
Présentation : Asif AHMED SHERAZI – Simon BENSIAM

5 décembre 2018

Exemple d’utilisation



Solution :

```
coup 1 : C3 va en F6
coup 2 : D7 va en D8
coup 3 : E3 va en E7
Echec et mat !!!
OU
coup 2 : E7 va en E5
coup 3 : E3 va en H3
Echec et mat !!!
OU
coup 2 : E7 va en E6
coup 3 : E3 va en A7
Echec et mat !!!
OU
coup 2 : E7 va en D6
coup 3 : E3 va en E7
Echec et mat !!!
OU
coup 2 : E7 va en F6
coup 3 : E3 va en E8
```

Fonctions principales utilisées :

void newp (plateau * p) : Permet de saisir de nouveaux problèmes.

void selecp (plateau p) : Permet de choisir le type de problèmes.

solution resop (plateau p, int coup, int nbc, int c) : fonction principale qui résout le problème.

Fonction newp()

```c
void newp(plateau *p){
    int r = 1;
    char r1;
    char r2;
    char coord[2];
    char type[2];
    int o,a; //Ordonnée et abscisse
    int t; //Type de pièce
    while(r){
        affp(*p);
        printf("Voulez-vous placez une nouvelle pièce ? (o/n)\n");
        scanf("%c",&r1);
        viderBuffer();
        while(r1 != 'o' && r1 != 'n'){
            printf("Réponse invalide (o/n)\n");
            scanf("%c",&r1);
            viderBuffer();
        }
        if (r1 == 'o'){
            printf("Ou voulez-vous placez votre pièce ? (Exemple : A1)\n");
            scanf("%s",coord);
            viderBuffer();
            while(coord[0] < 'A' || coord[0] > 'H' || coord[1] < '1' || coord[1] > '8'){
                if (coord[0] >= 'a' && coord[0] <= 'h'){
                    coord[0] += 'A' - 'a';
                    break;
```

```
        }
        printf("Les coordonnées que vous avez entré sont incorrectes. Ex:[A à H][1 à 8]\nVeuillez saisir de nouvelles coordonées\n");
        scanf("%s",coord);
        viderBuffer();
    }
    a = coord[0] - 'A';
    o = (coord[1] - '8') * -1;
    if (p->grille[o][a] != -1){
        printf("Il y a déjà une pièce sur cette case voulez-vous la remplacer ? (o/n)\n");
        scanf("%c",&r2);
        viderBuffer();
        while(r2 != 'o' && r2 != 'n'){
            printf("Réponse invalide (o/n)\n");
            scanf("%c",&r2);
            viderBuffer();
        }
        if(r2 == 'n'){
            continue;
        }
        p->grille[o][a] = -1;
        decomptepieces(p);
    }
    printf("Quel type de pièce ? (Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)\n");
    scanf("%s",type);
    viderBuffer();
    while(type[0] != 'R' && type[0] != 'D' && type[0] != 'T' && type[0] != 'F' && type[0] != 'C' && type[0] != 'P'){
        if(type[0] == 'r' || type[0] == 'd' || type[0] == 't' || type[0] == 'f' || type[0] == 'c' || type[0] == 'p'){
            type[0] += 'A' - 'a';
            break;
```

```
    }
    printf("Le type de pièce que vous avez entré est incorrect.\nEx:(Rn / Rb / Dn / Db / Tn / Tb / Fn / Fb / Cn / Cb / Pn / Pb)\nVeuillez sais
    scanf("%s",type);
    viderBuffer();
}
while(type[1] != 'n' && type[1] != 'b'){
    printf("La couleur de la pièce est incorrecte. (n pour noir / b pour blanc)\nVeuillez saisir une nouvelle couleur\n");
    char bn;
    scanf("%c",&bn);
    viderBuffer();
    type[1] = bn;
}
t = convert(type);


if (tropdepieces(*p,t)){
    printf("Il y a trop de pièces de ce type !!!\n\n");
    continue;
}

if (t == 10 || t == 11){
    if(o == 0 || o == 7){
        printf("Il ne peut pas y avoir de pion ici !!!\n\n");
        continue;
    }
}
if (t == 0){
    if( (p->grille[o - 1][a]     == 1    && coordvalide((o * 10 + a) - 10))  ||
        (p->grille[o - 1][a - 1] == 1    && coordvalide((o * 10 + a) - 11))  ||
        (p->grille[o - 1][a + 1] == 1    && coordvalide((o * 10 + a) - 9))   ||
        (p->grille[o][a - 1]     == 1    && coordvalide((o * 10 + a) - 1))   ||
        (p->grille[o][a + 1]     == 1    && coordvalide((o * 10 + a) + 1))   ||
        (p->grille[o + 1][a - 1] == 1    && coordvalide((o * 10 + a) + 9))   ||
        (p->grille[o + 1][a + 1] == 1    && coordvalide((o * 10 + a) + 11))  ||
        (p->grille[o + 1][a]     == 1    && coordvalide((o * 10 + a) + 10))) {

        printf("Il ne peut pas y avoir de roi blanc ici !!!\n\n");
        continue;
```

```
            }
        }
        if (t == 1){
            if( (p->grille[o - 1][a]     == 0    && coordvalide((o * 10 + a) - 10))  ||
                (p->grille[o - 1][a - 1] == 0    && coordvalide((o * 10 + a) - 11))  ||
                (p->grille[o - 1][a + 1] == 0    && coordvalide((o * 10 + a) - 9))   ||
                (p->grille[o][a - 1]     == 0    && coordvalide((o * 10 + a) - 1))   ||
                (p->grille[o][a + 1]     == 0    && coordvalide((o * 10 + a) + 1))   ||
                (p->grille[o + 1][a - 1] == 0    && coordvalide((o * 10 + a) + 9))   ||
                (p->grille[o + 1][a + 1] == 0    && coordvalide((o * 10 + a) + 11))  ||
                (p->grille[o + 1][a]     == 0    && coordvalide((o * 10 + a) + 10))) {

                printf("Il ne peut pas y avoir de roi noir ici !!!\n\n");
                continue;
            }
        }

        p->grille[o][a] = t;
        decomptepieces(p);
    }
    else{
        if(p->Nbp[0] == 0 || p->Nbp[1] == 0){
            printf("Il manque au moins un roi sur l'échiquier !!!\n\n");
            continue;
        }
        r = 0;
    }
  }
}
```

Fonction selecp()

```c
void selecp(plateau p){
    char c;
    printf("Qui doit jouer ?  (0 pour blanc / 1 pour noir)\n");
    scanf("%c",&c);
    viderBuffer();
    while(c != '0' && c != '1'){
        printf("Réponse invalide (0/1)\n");
        scanf("%c",&c);
        viderBuffer();
    }
    c -= '0';

    char nbc;
    printf("Echec et mat en combien de coups ?\n");
    scanf("%c",&nbc);
    viderBuffer();
    while(nbc < '1'){
        printf("Réponse invalide\n");
        scanf("%c",&nbc);
        viderBuffer();
    }
    nbc -= '0';

    solution s = resop(p, 1, nbc*2 -1, c);

    if(s.sol){
        printf("\n");
        affp(p);
        affichearbre(s.hist, 1);
    }
    else
        printf("Pas de solution...\n");
}
```

Structures utilisées :

```c
typedef struct plateau{
    int grille[8][8];
    int Nbp[12];
}plateau;

struct noeud {
    int coord;
    int coup;
    struct noeud * suiv[];
};
typedef struct noeud  noeud;
typedef struct noeud * arbre;

typedef struct solution{
    int sol;
    arbre hist;
}solution;
```