isec Engenharia

LICENCIATURA ENGENHARIA
INFORMÁTICA - CURSO EUROPEU

RELATÓRIO PROGRAMAÇÃO

PROJETO JOGO DO GALO

PROJETO REALIZADO POR:
ANTÓNIO DOMINGOS GONÇALVES PEDROSO - 2021132042

ANO LETIVO 2021/2022

ÍNDICE

I. INTRODUÇÃO

No presente documento irá ser registado todo o processo envolvido na realização do projeto do jogo do galo.

Os assuntos abordados neste relatório são baseados no conhecimento de linguagem C que eu tenho, maior parte do mesmo aprendido nas aulas.

O projeto foi constituído por vários componentes sustentados por uma ideia inicial, a criação de um mega jogo do galo no contexto da unidade curricular de Programação.

O projeto é composto por vários ficheiros, os quais contem funções e métodos que servem para o auxílio do ficheiro principal.

No decorrer deste documento será possível acompanhar todo o trabalho que tive ao longo deste projeto, dificuldades enfrentadas e descrição de todas as componentes deste projeto.

II. SOFTWARES UTILIZADOS

II.I IDE

- CLion 2021.3.2

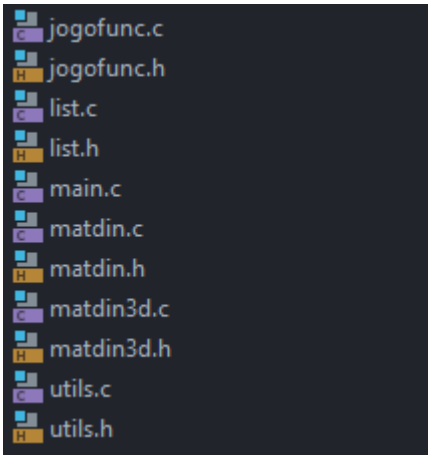
II.II COMPILADOR

- MinGW w64 9.0

III. DESENVOLVIMENTO DO PROJETO

O desenvolvimento do projeto repartiu-se em 6 parte:

- utils – Ficheiro que contem as funções de utilidades
- matdin – Ficheiro para gerir matrizes 2d
- matdin3d – Ficheiro para gerir matrizes 3d
- jogofunc – Ficheiro que contem as principais funções do jogo
- list – Ficheiro para gerir as listas ligadas
- main – Ficheiro principal



III.I UTILS

Este ficheiro foi fornecido pelo docente responsável da disciplina.

```c
// Inicializa o gerador de numeros aleatorios.
// Esta funcao deve ser chamada apenas uma vez no inicio da execucao do programa
void initRandom(){
    srand(time(NULL));
}

//Devolve um valor inteiro aleatorio distribuido uniformemente entre [a, b]
int intUniformRnd(int a, int b){
    return a + rand()%(b-a+1);
}
```

III.II MATDIN

Este ficheiro foi fornecido pelo docente responsável da disciplina.

```c
// Liberta uma matriz dinâmica de caracteres com nLin linhas
void libertaMat(char** p, int nLin){

    int i;

    for(i=0; i<nLin; i++)
        free(p[i]);
    free(p);
}

// Cria uma matriz dinâmica de caracteres  com nLin linhas e nCol colunas
// Devolve endereço inicial da matriz
char** criaMat(int nLin, int nCol){
    char **p = NULL;
    int i, j;

    p = malloc(sizeof(char*) * nLin);
    if(p == NULL)
        return NULL;

    for(i=0; i<nLin; i++){
        p[i] = malloc(sizeof(char) * nCol);
        if(p[i] == NULL){
            libertaMat(p, i-1);
            return NULL;
        }
        for(j=0; j<nCol; j++)
            p[i][j] = '-';
    }
    return p;
}
```

```c
// Coloca o caracter c na posição (x, y) de uma matriz dinâmica de caracteres
void setPos(char **p, int x, int y, char c){
    p[x][y] = c;
}

// Obtém o caracter armazenado na posição (x, y) de uma matriz de caracteres
char getPos(char **p, int x, int y){
    return p[x][y];
}
```

III.III MATDIN3D

Este ficheiro é uma ligeira adaptação do ficheiro matdin.

```c
// IMPRIME O JOGO NO ECRA
void mostraJogo(char ***p){
    printf("\nTabuleiro: \n");
    int aux = 0;
    int jIni = 0;
    int jFim = 3;

    for(int i=0; i<3; i++){
        if(aux == 9){
            return;
        }
        printf("\t");
        for(int j=jIni; j<jFim; j++){
            for(int k = 0; k<3; k++){
                printf("%c ", p[j][i][k]);
            }
            printf("\t");
        }
        if(i == 2){
            i = -1;
            jIni = jFim;
            jFim += 3;
            printf("\n");
        }
        printf("\n");
        aux++;
    }
}

// LIBERTA A MATRIZ 3D
void libertaMat3D(char*** nTab, int nLin, int nCol){
    for(int i=0; i<nLin; i++){
        for(int j=0; j<nCol; j++){
            free(nTab[i][j]);
        }
        free(nTab[i]);
    }
    free(nTab);
}
```

```c
// CRIA UMA MATRIZ 3D COM O NLIN E NCOL;
char*** criaMat3D(int nTab, int nLin, int nCol){
    char ***p = NULL;

    p = malloc(sizeof(char **) * nTab);
    if(p == NULL)
        return NULL;

    for(int i=0; i<nTab; i++){
        p[i] = malloc(sizeof(char*) * nLin);
        for(int j=0; j<nLin; j++){
            p[i][j] = malloc(sizeof(char) * nCol);

            if(p[i][j] == NULL){
                libertaMat3D(p, i-1, j-1);
                return NULL;
            }

            for(int a=0; a<nCol; a++){
                p[i][j][a] = '-';
            }
        }
    }

    return p;
}

// MUDA O VALOR DE UMA POSICAO PASSADA POR PARAMETROS NA MATRIZ 3D
void setPos3D(char ***p, int tab, int x, int y, char c){
    p[tab][x][y] = c;
}

// OBTEM O VALOR DE UMA POSICAO PASSADA POR PARAMETROS DA MATRIZ 3D
char getPos3D(char ***p, int tab, int x, int y){
    return p[tab][x][y];
}
```

III.IV JOGOFUNC

```c
// VERIFICA SE ALGUM JOGADOR GANHOU O JOGO, RETORNA x PARA O JOGADOR1,
// o PARA O JOGADOR 2, d EM CASO DE EMPATE E n NO CASO DO JOGO AINDA NÃO ESTAR ACABADO
char checkWinner(char **p, char ***tab){
    if(p[0][0] != '-' && p[0][0] == p[0][1] && p[0][0] == p[0][2]){ return getPos(p, 0, 0); }
    if(p[1][0] != '-' && p[1][0] == p[1][1] && p[1][0] == p[1][2]){ return getPos(p, 1, 0); }
    if(p[2][0] != '-' && p[2][0] == p[2][1] && p[2][0] == p[2][2]){ return getPos(p, 2, 0); }

    if(p[0][0] != '-' && p[0][0] == p[1][0] && p[0][0] == p[2][0]){ return getPos(p, 0, 0); }
    if(p[0][1] != '-' && p[0][1] == p[1][1] && p[0][1] == p[2][1]){ return getPos(p, 0, 1); }
    if(p[0][2] != '-' && p[0][2] == p[1][2] && p[0][2] == p[2][2]){ return getPos(p, 0, 2); }

    if(p[0][0] != '-' && p[0][0] == p[1][1] && p[0][0] == p[2][2]){ return getPos(p, 0, 0); }
    if(p[2][0] != '-' && p[2][0] == p[1][1] && p[2][0] == p[0][2]){ return getPos(p, 2, 0); }

    int total = 0;
    for(int i=0; i<9; i++){
        for(int k=0; k<3; k++){
            for(int j=0; j<3; j++){
                if(getPos3D(tab, i, k, j) == '-'){
                    total++;
                }
            }
        }
    }
    if(total == 81){
        return 'd';
    }

    return 'n';
}
```

```c
// FUNCAO PARA OBTER O TABULEIRO SELECIONADO PARA A PROXIMA JOGADA
int updateTab(char **p, int x, int y){
    int val = 0;
    if(p[x][y] == '-'){
        if(x == 0 && y == 0){ return 0; }
        else if(x == 0 && y == 1){ return 1; }
        else if(x == 0 && y == 2){ return 2; }
        else if(x == 1 && y == 0){ return 3; }
        else if(x == 1 && y == 1){ return 4; }
        else if(x == 1 && y == 2){ return 5; }
        else if(x == 2 && y == 0){ return 6; }
        else if(x == 2 && y == 1){ return 7; }
        else if(x == 2 && y == 2){ return 8; }
    }else{
        for(int i=0; i<3; i++){
            for(int k=0; k<3; k++){
                if(p[i][k] == '-'){
                    printf("Tabuleiro indisponivel, o proximo tabuleiro sera o tabuleiro numero %d\n", val+1);
                    return val;
                }
                val++;
            }
        }

    }
    return val;
}
```

```
// ATUALIZA O TABULEIRO GRANDE DE ACORDO COM OS TABULEIROS PEQUENOS
void updateTabGrande(char ***tab, char **p){
    int curr = 0;
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            if(p[i][j] == '-'){
                if(getPos3D(tab, curr, 0, 0) != '-' && getPos3D(tab, curr, 0, 1) != '-' && getPos3D(tab, curr, 0, 2) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 0)); }

                if(getPos3D(tab, curr, 1, 0) != '-' && getPos3D(tab, curr, 1, 1) != '-' && getPos3D(tab, curr, 1, 2) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 1, 0)); }

                if(getPos3D(tab, curr, 2, 0) != '-' && getPos3D(tab, curr, 2, 1) != '-' && getPos3D(tab, curr, 2, 2) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 2, 0)); }


                if(getPos3D(tab, curr, 0, 0) != '-' && getPos3D(tab, curr, 1, 0) != '-' && getPos3D(tab, curr, 2, 0) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 0)); }

                if(getPos3D(tab, curr, 0, 1) != '-' && getPos3D(tab, curr, 1, 1) != '-' && getPos3D(tab, curr, 2, 1) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 1)); }

                if(getPos3D(tab, curr, 0, 2) != '-' && getPos3D(tab, curr, 1, 2) != '-' && getPos3D(tab, curr, 2, 2) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 2)); }

                if(getPos3D(tab, curr, 0, 0) != '-' && getPos3D(tab, curr, 1, 1) != '-' && getPos3D(tab, curr, 2, 2) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 0)); }

                if(getPos3D(tab, curr, 2, 2) != '-' && getPos3D(tab, curr, 1, 1) != '-' && getPos3D(tab, curr, 0, 3) != '-'){
                    setPos(p, i, j, getPos3D(tab, curr, 0, 0)); }
            }
            curr++;
        }
    }
}
```

```
// FAZ UMA JOGADA ALEATORIA NO TABULEIRO SELECIONADO
void randomPlay(char ***tab, int curr, int* x, int* y){
    initRandom();
    int a,b;
    do{
        a = intUniformRnd(0,2);
        b = intUniformRnd(0,2);
        if(getPos3D(tab, curr, a, b) == '-'){
            *x = a;
            *y = b;
        }
    }while(getPos3D(tab, curr, a, b) != '-');
}
```

III.V LIST

```c
// VERIFICA SE A LISTA ESTÁ VAZIA, DEVOLVE 1 NO CASO DE ESTAR VAZIA
int lista_vazia(pjogada p){
    if(p == NULL){
        return 1;
    }else{
        return 0;
    }
}

// OBTEM O JOGADOR DA ULTIMA JOGADA NA LISTA
char get_last_player(pjogada p){
    if(p->njogador == 1){
        return 'x';
    }else{
        return 'o';
    }
}

// OBTEM A COORDENADA X DA ULTIMA JOGADA NA LISTA
int get_last_x(pjogada p){
    return p->cox;
}

// OBTEM A COORDENADA Y DA ULTIMA JOGADA NA LISTA
int get_last_y(pjogada p){
    return p->coy;
}

// OBTEM U NUMERO DA ULTIMA JOGADA NA LISTA
int get_last_jogada(pjogada p){
    return p->njogada;
}
```

```c
// ATUALIZA O TABULEIRO A PARTIR DOS VALORES CONTIDOS NA LISTA
void atualiza_tabuleiro(pjogada p, char*** tab){
    char bac;
    while(p != NULL){
        if(p->njogador == 1){
            bac = 'x';
        }else{
            bac = 'o';
        }
        setPos3D(tab, p->ntab, p->cox, p->coy, bac);
        p = p->prox;
    }
}

// IMPRIME AS ULTIMAS X JOGADAS PASSADAS NO PARAMETRO NJOGADAS NA LISTA
void mostra_infoX(pjogada p, int njogadas){
    int nig = 0;
    while(p != NULL && nig<njogadas){
        printf("Jogada: %d\tJogador: %d\tNum Tabuleiro: %d\tLinha: %d\tColuna: %d\n", p->njogada, p->njogador, p->ntab+1, p->cox+1, p->coy+1);
        p = p->prox;
        nig++;
    }
}

// INSERE UM NOVO REGISTO NA LISTA
pjogada insere_inicio(pjogada p, int njogada, int njogador, int ntab, int cox, int coy){
    pjogada novo;

    if((novo = malloc(sizeof(jogada))) == NULL){
        printf("Erro na alocacao de memoria\n");
    }else{
        preenche(novo, njogada, njogador, ntab, cox, coy);
        novo->prox = p;
        p = novo;
    }

    return p;
}
```

```c
// FUNÇÃO PARA AJUDAR NA INSERCAO DE UM NOVO REGISTO NA LISTA
void preenche(pjogada p, int njogada, int njogador, int ntab, int cox, int coy){
    p->njogada = njogada;
    p->njogador = njogador;
    p->ntab = ntab;
    p->cox = cox;
    p->coy = coy;
    p->prox = NULL;
}

// GRAVA A LISTA PARA UM FICHEIRO, RETORNA 1 EM CASO DE SUCESSO
int grava_lista(pjogada lista, char* nomeF){
    FILE *f;

    f = fopen(nomeF, "wb");
    if(f == NULL){
        return 0;
    }else{
        while(lista != NULL){
            fprintf(f, "%d # %d # %d # %d # %d\n", lista->njogada, lista->njogador, lista->ntab, lista->cox, lista->coy);
            lista = lista->prox;
        }
        fclose(f);
    }
    return 1;
}

// LIBERTA O ESPAÇO EM MEMORIA USADO PELA LISTA
void liberta_lista(pjogada p){
    pjogada aux;

    while(p != NULL){
        aux = p;
        p = p->prox;
        free(aux);
    }
}
```

```c
// OBTEM A LISTA A PARTIR DE UM FICHEIRO
pjogada recupera_lista(char *nomeF){
    pjogada novo, aux, lista=NULL;
    FILE *f;
    jogada j;

    f = fopen(nomeF, "rb");

    if(f == NULL){
        return NULL;
    }

    while(fscanf(f, "%d # %d # %d # %d # %d", &(j.njogada), &(j.njogador), &(j.ntab), &(j.cox), &(j.coy)) != EOF){
        j.prox = NULL;
        novo = malloc(sizeof(jogada));
        if(novo == NULL){
            fclose(f);
            liberta_lista(lista);
            return NULL;
        }

        *novo = j;

        if(lista == NULL){
            lista = novo;
        }else{
            aux = lista;
            while(aux->prox != NULL){
                aux = aux->prox;
            }
            aux->prox = novo;
        }
    }
    fclose(f);

    return lista;
}
```

III.VI MAIN

```c
int main() {
    initRandom();

    char** tabGrande = criaMat(3,3);
    char*** tabPequeno = criaMat3D(9,3,3);
    int auxx, auxy, tabaux, prog, currentTab, temp = 2, scanie, scaniej, escolhamodo, escolhatabinicial, numjogadas = 1, turnojog = 1;
    char atual = 'x';

    pjogada lista = NULL;

    lista = recupera_lista("jogadas.bin");

    printf("Bem-vindo ao Jogo do Galo.\n");
    printf("Os tabuleiros serao designados da seguinte forma:\n\n");
    printf(" Tab1 \t Tab2 \t Tab3 \n Tab4 \t Tab5 \t Tab6 \n Tab7 \t Tab8 \t Tab9\n\n");
    printf("As posicoes serao designadas da seguinte forma:\n\n");
    printf(" L1C1 \t L1C2 \t L1C3 \n L2C1 \t L2C2 \t L2C3 \n L3C1 \t L3C2 \t L3C3\n\n");


    if(lista_vazia(lista) == 0){
        printf("Deseja continuar o jogo anteriormente gravado?\n\t1- Sim\n\t2- Nao\nIntroduza a escolha:");
        scanf("%d", &temp);

        if(temp == 1){
            atualiza_tabuleiro(lista, tabPequeno);
            updateTabGrande(tabPequeno, tabGrande);
            atual = get_last_player(lista);
            auxx = get_last_x(lista);
            auxy = get_last_y(lista);
            numjogadas = get_last_jogada(lista) + 1;
            currentTab = updateTab(tabGrande,auxx,auxy);

            mostraJogo(tabPequeno);
        }else{
            lista = NULL;
        }
    }
```

```c
    printf("Escolha o tipo de jogo:\n\t1- Jogo entre 2 jogadores humanos.\n\t2- Jogo entre um humano e um jogador automatico.\nIntroduza a escolha:");
    do{
        scanf("%d", &escolhamodo);
        if(escolhamodo < 1 || escolhamodo > 2){
            printf("Escolha invalida.\nIntroduza a escolha:");
        }
    }while(escolhamodo < 1 || escolhamodo > 2);

    if(temp == 2){
        printf("\nEscolha se quer escolher o tabuleiro inicial:\n\t1- Escolher tabuleiro inicial.\n\t2- Tabuleiro inicial aleatorio.\nIntroduza a escolha:");
        do{
            scanf("%d", &escolhatabinicial);
            if(escolhatabinicial < 1 || escolhatabinicial > 2){
                printf("Escolha invalida.\nIntroduza a escolha:");
            }
        }while(escolhatabinicial < 1 || escolhatabinicial > 2);

        if(escolhatabinicial == 1){
            printf("Escolha o numero do tabuleiro inicial:");
            scanf("%d", &currentTab);
        }else{
            currentTab = intUniformRnd(0,8);
        }
    }
```

```
do{
    printf("\nTabuleiro numero %d selecionado\n", currentTab+1);
    printf("Turno do jogador %d\n", turnojog);

    do{
        if(numjogadas >1 && turnojog == 1){
            printf("Guardar progresso e sair?\n\t1- Sim\n\t2- Nao\nEscolha:");
            scanf("%d", &prog);
            if(prog == 1){
                int b = grava_lista(lista, "jogadas.bin");
                if(b == 1){
                    printf("Progesso guardado. A sair...");
                }else{
                    printf("Ocorreu um erro na gravacao. A sair...");
                }
                return 1;
            }
        }

        if(numjogadas >1){
            printf("\nQuer ver as jogadas anteriores?\n\t1- Sim\n\t2- Nao\nEscolha:");
            scanf("%d", &scanie);

            if(scanie == 1){
                do{
                    printf("Quantas jogadas quer ver? (1 a 10)\nNum de jogadas a ver:");
                    scanf("%d", &scaniej);
                    if(scaniej < 1 || scaniej > 10){
                        printf("Numero invalido.\nIntroduza um numero entre 1 e 10:");
                    }
                }while(scaniej < 1 || scaniej > 10);

                printf("\n");
                mostra_infoX(lista, scaniej);
            }
        }
```

```
        printf("Linha (entre 1 e 3):");
        do{
            scanf("%d", &auxx);
            if(auxx < 1 || auxx > 3) {
                printf("Linha invalida.\nIntroduza linha:");
            }
        }while(auxx < 1 || auxx > 3);

        printf("Coluna (entre 1 e 3):");
        do{
            scanf("%d", &auxy);
            if(auxy < 1 || auxy > 3){
                printf("Coluna invalida.\nIntroduza coluna:");
            }
        }while(auxy < 1 || auxy > 3);

        auxx--;
        auxy--;

        if(getPos3D(tabPequeno,currentTab, auxx, auxy) != '-'){
            printf("Coordenadas invalidas.\n");
            mostraJogo(tabPequeno);
        }
    }while(getPos3D(tabPequeno,currentTab, auxx, auxy) != '-');

    printf("\n");

    setPos3D(tabPequeno, currentTab, auxx, auxy, atual);
    updateTabGrande(tabPequeno, tabGrande);

    tabaux = currentTab;
    currentTab = updateTab(tabGrande,auxx,auxy);

    if(getPos3D(tabPequeno,currentTab, auxx, auxy) == '-'){
        lista = insere_inicio(lista, numjogadas, turnojog, tabaux, auxx, auxy);
        numjogadas++;
    }
```

```
        if(escolhamodo == 2){
            tabaux = currentTab;
            randomPlay(tabPequeno, currentTab, &auxx, &auxy);
            setPos3D(tabPequeno, currentTab, auxx, auxy, 'o');
            updateTabGrande(tabPequeno, tabGrande);

            lista = insere_inicio(lista, numjogadas, 2, tabaux, auxx, auxy);
            numjogadas++;

            currentTab = updateTab(tabGrande,auxx,auxy);
        }

        if(escolhamodo == 1){
            if(turnojog == 1){
                turnojog = 2;
                atual = 'o';
            }else{
                turnojog = 1;
                atual = 'x';
            }
        }

        mostraJogo(tabPequeno);
    }while(checkWinner(tabGrande, tabPequeno) == 'n');

    char winner = (checkWinner(tabGrande, tabPequeno));

    if(winner == 'x'){
        printf("Ganhou o jogador 1 (%c)", winner);
    }else if(winner == 'o'){
        printf("Ganhou o jogador 2 (%c)", winner);
    }else{
        printf("Empate");
    }

    return 0;
}
```

IV. JUSTIFICAÇÃO DE ESCOLHAS

Optei por usar 2 matrizes neste projeto, pois achei mais otimizado e acessível em termos de manipulação de informação. A primeira matriz cujo nome “tabGrande” que é a matriz 2d que contém a informação sobre quem venceu em cada tabuleiro e matriz com o nome “tabPequeno” que é a matriz 3d que contém toda a informação de todos os tabuleiros pequenos dentro do tabuleiro grande.

No início é perguntado ao jogador se ele quer continuar o jogo guardado no ficheiro (no caso do mesmo existir), se o jogador disser sim pode posteriormente escolher se pretende jogar contra outro jogador ou sozinho.

Para escolha do tabuleiro inicial eu optei por dar ao utilizador a opção de escolher se quer randomizar o primeiro tabuleiro ou se quer escolher o mesmo.

CONCLUSÃO

Em tese isto foi um projeto que agregou todos os meus conhecimentos aprendidos ao longo desta UC, no entanto não ficou por aí, eu acabei por tentar fazer um projeto mais pessoal onde pudesse colocar as minhas capacidades adquiridas à prova.

Desde que comecei este projeto, ele foi a minha maior prioridade, para mim criar um projeto que me deixasse feliz não bastava, eu queria ir mais além e tentar desafiar as minhas capacidades cognitivas e fazer um projeto com que eu me orgulhasse bastante.

Com este projeto, adquiri vários conhecimentos, conhecimentos esses que não possuía antes de começar a realização do mesmo.

Alcancei todos os objetivos que tinha em mente e foram surgindo com o passar do tempo.

Foi, com certeza, fundamental a realização deste projeto pois pude conciliar as minhas capacidades de programação com o meu desenvolvimento académico.