ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Relatório
Laboratórios de Programação

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

2017/2018

Grupo 60
8160445 – Daniel Castro
8170265– Pedro Lopes

1. Tema

O jogo da torre de Hanói consiste em uma base com o número de 3 pinos em quais estão dispostos um número de, sendo que 3 é o mínimo de discos, que estão dispostos por ordem crescente de diâmetro. O objetivo consiste em passar todos os discos de um pino para outro qualquer, usando um dos pinos como auxiliar, de maneira que um disco maior nunca fique em cima de outro menor em nenhuma situação, no menor número de movimentos possíveis.

2. Funcionalidades propostas

O jogo da torre de Hanói terá um sistema de dificuldade escolhido pelo próprio jogador ao escolher o número de discos que pretende, sendo 3 o mínimo e o mais fácil.

Será implementado um sistema de pontuação em que regista o número de movimentos e tempo, sendo o objetivo terminar o jogo no menor número de movimentos e menor tempo. Estas pontuações serão associadas a um perfil criado pelo jogador, no menu ou senão existir no fim do jogo terá a opção para o criar.

3.Estrutura analítica do projeto

1º. Subdivisão do projeto:

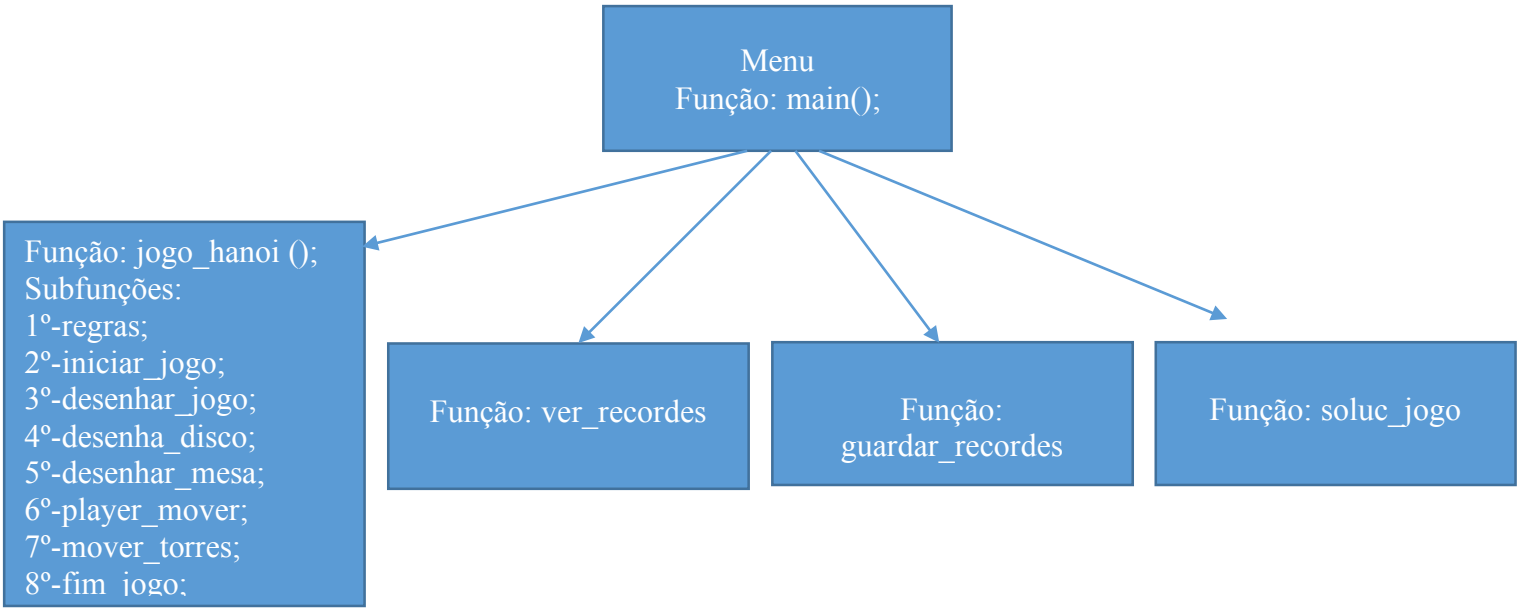
Podemos subdividir este projeto em três partes: o jogo, recordes e simulação de jogo

O jogo subdivide-se também em cinco partes: inicialização, desenhar, jogada do jogador, mover discos de acordo com a jogada do jogador e por fim a verificação do fim do jogo (caso o jogo continua ou termina).

Recordes divide-se em duas partes ver os recordes e guardar os recordes.

Simulação de jogo é uma simulação visual com as respetivas soluções;

2º. Planeamento do projeto e Organização do projeto



A função main subdivide-se em três funções: jogo_hanoi, ver_recordes e soluc_jogo;

Jogo_hanoi é que se divide em sete funções: regras, iniciar_jogo, desenhar_jogo, player_mover, mover_torres, fim_jogo e por fim guardar_recordes.

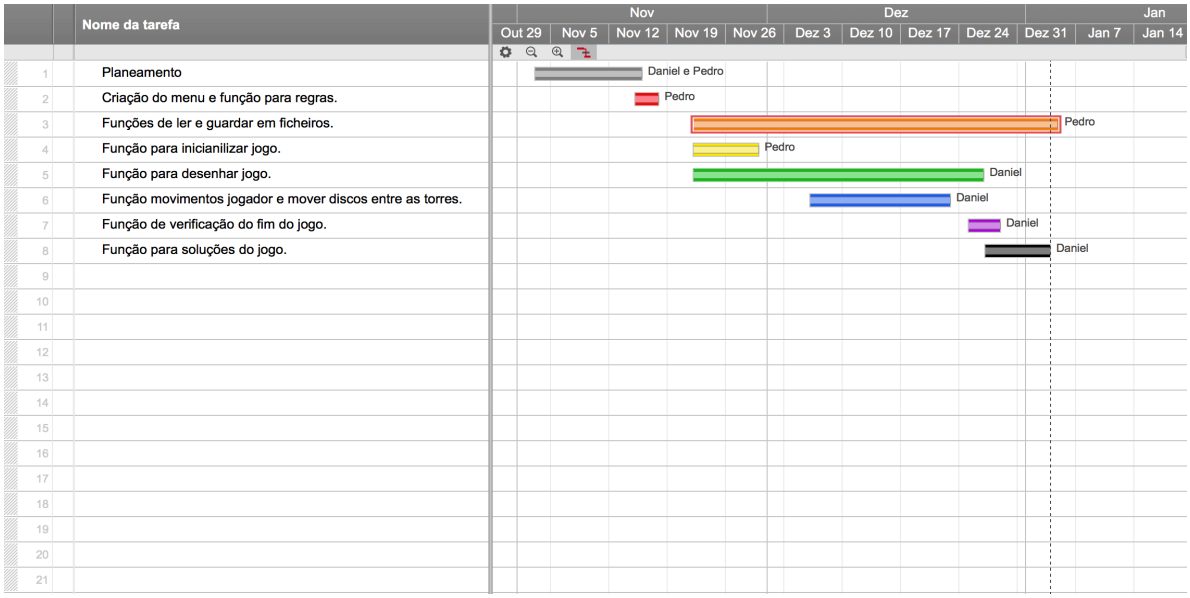
Responsáveis:

Daniel Castro (8160445): desenhar_jogo, desenhar_mesa, desenhar_disco, player_mover, mover_torres, fim_jogo, soluc_jogo;

Pedro Lopes (8170265): main("menu"), regras, iniciar_jogo, guardar_recordes, ver_recordes;

Bibliotecas: stdio.h, stdlib.h e time.h.

3º. Planeamento temporal (Diagrama de Gantt)



Ver datas(**link**): https://app.smartsheet.com/b/publish?EQBCT=6bd2fbd1eb2840e5bb6b8bdaba9df1eb.

4.Funcionalidades implementadas

1) Função jogo_hanoi();

```c
void jogo_hanoi(PLAYER *p, int contador, int n_discos) {
    int n, t;
    time_t start, end;

    iniciar_jogo(n_discos);

    start = time(0);

    do {
        desenhar_jogo();
        do {
            player_mover();
        } while (m.origem == -1 && m.destino == -1);

        if (mover_torres(m.origem, m.destino) == 1) {
            if (fim_jogo() == 1) {
                n = 1;
                p[contador].moves++;
                desenhar_jogo();
                printf("\nFim jogo");
            } else {
                n = 0;
                p[contador].moves++;
            }
        }
    } while (n == 0);

    end = time(0);

    p[contador].tempo = end - start;

}
```

Esta função começa com a variável **start** que inicia a contagem do tempo do jogador recorrendo à função **time**, depois passa para o primeiro ciclo do-while que só terminar se a variável de condição "n" for igual a 0 e a seguir à função **desenhar_jogo** segue um terceiro ciclo do-while responsável por continuar a executar a função **player_mover** até que o jogador escolha as opções corretas para continuar a jogar. Assim para a primeira condição em que a função **mover_torres,** se devolver o valor 1 passa para a próxima condição que em que a função **fim_jogo** se devolver o valor 1, permite terminar o jogo e o segundo ciclo, porque n passa a ser igual a 1, caso contrario o jogo continuava, pois n teria o valor de 0. Assim **end** termina de contar o tempo, utilizando mais uma vez a função **time**, e guardar no variável tempo da estrutura PLAYER.

2) Função mover_torres();

```
73  int mover_torres(int origem, int destino) {
74      int i, n, k;
75
76      for (i = h.tam_max_discos - 1; i >= 0; i--) {
77
78
79          if (h.torre[origem].discos[i] != 0) {
80              n = h.torre[origem].discos[i];
81              h.torre[origem].discos[i] = 0;
82              k = i;
83              i = 0;
84          } else
85              n = 0;
86      }
87
88      for (i = 0; i < h.tam_max_discos; i++) {
89          if (n > h.torre[destino].discos[i] && h.torre[destino].discos[i] != 0) {
90              printf("Movimento impossivel");
91              h.torre[origem].discos[k] = n;
92
93              return 0;
94
95          } else
96
97              if (h.torre[destino].discos[i] == 0) {
98              h.torre[destino].discos[i] = n;
99              i = h.tam_max_discos;
100
101             return 1;
102         }
103
104
105
106     }
107
108 }
```

A função executa em primeiro lugar o primeiro ciclo for, que é responsável por procurar o disco do topo da torre selecionada, pelo jogador, para ser retirado. Caso o jogador não respeite a regra de não mover um disco de maior diâmetro para uma torre em que o disco do topo é de menor diâmetro em relação ao disco retirado, a variável k é responsável por guardar a posição na torre em que foi retirado.

De seguida um segundo ciclo for que tem função de percorrer a torre onde vai ser colocado o disco retirado. Existindo duas condições, sendo a primeira responsável por confirmar-se o disco retirado é de maior diâmetro que o disco do topo da torre a que tem destino, e responsável por voltar a colocar de volta o disco retirado, caso esta se confirme ser verdade. A segunda só atua se a primeira for falsa, tendo função de colocar o disco na torre a que tem destino.

3) Função ver_recordes();

```c
int ver_recordes(PLAYER *p){
    char c;
    int n_discos, i, lines = 0;
    do {
        printf("\nNº de discos(MIN:3 MÁX:8):");
        scanf("%d", &n_discos);
    } while (n_discos < 3 || n_discos > 8);

    FILE *fp;

    if (n_discos == 3) {
        fp = fopen("file_n3.txt", "r");

    } else
        if (n_discos == 4) {
        fp = fopen("file_n4.txt", "r");
    } else
        if (n_discos == 5) {
        fp = fopen("file_n5.txt", "r");
    } else
        if (n_discos == 6) {
        fp = fopen("file_n6.txt", "r");
    } else
        if (n_discos == 7) {
        fp = fopen("file_n7.txt", "r");
    } else
        if (n_discos == 8) {
        fp = fopen("file_n8.txt", "r");
    }
    if (fp == NULL) {
        perror("Erro a abrir o ficheiro");
        return -1;
    }else{

    while ((c = fgetc(fp)) != EOF) {
        if (c == '\n') {
            lines++;
        }
    }
     rewind(fp);

     if (lines == 0)printf("\nNão existem jogadores nos recordes.");
     else {

         for (i = 0; i < lines; i++) {
             fscanf(fp, " %30[^\t]", p[i].name);
             fscanf(fp, "%d[^\t]", &p[i].moves);
             fscanf(fp, "%f[^\n]", &p[i].tempo);
         }


         for (i = 0; i < lines; i++) {
             printf("\n==================================");
             printf("\nPLAYER %d", i + 1);
             printf("\nNome:%s", p[i].name);
             printf("\nMovimetos:%d", p[i].moves);
             printf("\nTempo:%.2fs", p[i].tempo);
             printf("\n==================================");
         }

         fclose(fp);

     }
     return 0;
     }
}
```

No código da função a cima, inicia-se por pedir ao jogador em um ciclo while o n_discos(valor compreendido entre 3 e 8), de seguida com o valor de n_discos é comparado com os valores compreendidos e assim abre-se o ficheiro correto com o formato "r" de leitura , com a função **fopen**.

Um segundo ciclo while é executado com objetivo de contar o número de jogadores já presentes no ficheiro.

De seguida a função **rewind**, que é necessária para voltar a ler o ficheiro desde o início, assim caso a condição "lines==0" seja falsa inicia-se o primeiro ciclo for que percorre o ficheiro para guardar o nome, a pontuação e o tempo do respetivo jogador pela mesma ordem que do ficheiro, na estrutura PLAYER.

Por fim é executado também outro ciclo for que imprime todos os jogadores do ficheiro que foram salvos na estrutura PLAYER. Terminado com a função **fclose** que acaba com a stream do ficheiro, fechando-o.

4) Função guardar_recordes();

```
401 void guardar_recordes(PLAYER *p, int contador, int n_discos) {
402     char name[30];
403     FILE *fp;
404
405
406     if (n_discos == 3) {
407         fp = fopen("file_n3.txt", "a");
408     } else
409         if (n_discos == 4) {
410         fp = fopen("file_n4.txt", "a");
411     } else
412         if (n_discos == 5) {
413         fp = fopen("file_n5.txt", "a");
414     } else
415         if (n_discos == 6) {
416         fp = fopen("file_n6.txt", "a");
417     } else
418         if (n_discos == 7) {
419         fp = fopen("file_n7.txt", "a");
420     } else
421         if (n_discos == 8) {
422         fp = fopen("file_n8.txt", "a");
423     }
424
425     printf("\nTempo:%.2f s.\n", p[contador].tempo);
426     printf("\nPontuação(nº de movimentos):%d", p[contador].moves);
427
428     printf("\nNome:");
429     scanf(" %[^\n]s", p[contador].name);
430
431     fprintf(fp, " %s", p[contador].name);
432     fprintf(fp, "\t %d", p[contador].moves);
433     fprintf(fp, " \t %.2f \n", p[contador].tempo);
434
435     fclose(fp);
436
437     return;
438 }
```

A função **guardar_recordes** arranca com a comparação da variável que recebe com o valor de disco, que o jogo Hanoi foi iniciado, comparando-o com os pretendidos (3-8) para que possa abrir o ficheiro correto, com a função **fopen** e o modo de escrever sem apagar o conteúdo do ficheiro" a". Seguidamente imprime os resultados do fim de jogo, guardados na estrutura PLAYER e de seguida pede o nome ao jogador e o guarda na estrutura PLAYER.

Finalmente guarda a estrutura PLAYER no ficheiro e termina a stream do ficheiro com a função **fclose**.

5.Conclusão

Neste trabalho conseguimos realizar com sucesso as funcionalidades propostas inicialmente, ainda implementando uma funcionalidade de obter as soluções de jogo, com o número de discos pretendido. Mesmo assim, se tivéssemos mais tempo pretendíamos melhorar/adicionar ao nosso projeto funcionalidade como: melhora dos perfis de jogador, de modo a não repetir o mesmo jogador nos recordes; criação de um top 10 dos melhores jogadores, de acordo com os movimentos e tempo; e por fim melhorar as soluções de jogo adicionar os mesmos gráficos utilizados durante o jogo Torres de Hanói.

Concluindo, este projeto permitiu que as nossas capacidades de programador e de autonomia se desenvolvessem, que são essenciais para nossa continuação no curso de licenciatura de engenharia informática.