DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Programação
MEEC (2018/2019 – 2º Sem)

Projecto Intermédio

Grupo: 3

Aluno nº: 93030

Nome: Bruno Miguel Da Silva Cebola

Aluno nº: 93065

Nome: Frederico Maria de Almeida Santos

1. Introdução

Em termos gerais, o desafio proposto consiste em criar um programa que permita até 4 jogadores jogar uma versão modificada do clássico *Mastermind*. O programa a implementar tem de seguir uma estrutura geral descrita no enunciado: primeiramente, deve perguntar ao utilizador qual a configuração que o jogo deve ter (numero de jogadores e respetivos nomes, numero de jogos, dimensão e complexidade da chave, limite máximo de tempo e jogadas por jogo); de seguida, deve executar a sequencia de jogos para cada jogador; finalmente apresenta as estatísticas em três categorias: vencedor do torneio (mais jogos ganhos, desempata pelo tempo médio de jogo); jogo mais rápido (chave acertada em menos tempo, desempata pelo numero de jogadas); jogo mais curto (chave acertada em menos jogadas, desempata pelo tempo de jogo).

2. Implementação

Dado que o enunciado promove a tomada de decisões, a nossa interpretação e abordagem ao desafio proposto definiram a nossa implementação das seguintes formas:

-Consideramos este programa como sendo um torneio e como tal as regras e configurações são idênticas para todos os participantes;

-Os nomes dos jogadores podem aceitar todo o tipo de caracteres (incluindo caracteres especiais, espaços, etc.) desde que não ultrapasse o limite máximo imposto, tal como acontece com os *nicknames* dos jogos atuais;

-Durante o jogo, qualquer jogada inválida (utilização de cores não disponíveis, outras letras, números, ou caracteres especiais) é considerada como um erro de atenção não sendo contabilizada como uma tentativa, no entanto o tempo do jogo continua a correr normalmente;

-Como exigido no enunciado, as cores serão representadas por letras maiúsculas (quando impressas no terminal). No entanto, o programa aceita a utilização tanto de letras maiúsculas como minúsculas, quando o utilizador realiza as jogadas;

-Caso nenhum jogador tenha ganho um único jogo, as estatísticas apresentam uma mensagem explicando que não há vencedores em nenhuma categoria;

-É comum os jogadores não quererem apenas saber quem ganhou em certas categorias, mas poderem avaliar e comparar a sua performance individual entre eles, por isso incluímos a opção de apresentar um resumo da performance de cada jogador, após as estatísticas;

-Todos os "*inputs*" são verificados, pelo que em caso algum o programa entra em "*loops*" infinitos, fecha, bloqueia, ou comporta-se de forma anormal devido a input inesperado. Por outras palavras, o utilizador pode inserir qualquer input, e caso se engane apenas aparece uma mensagem de erro. Inputs do tipo " 2" ou "2 " (espaços antes ou depois) são considerados como inválidos pois podem ser erros dos utilizadores.

-O programa tem uma interface simples e o mais amiga do utilizador possível. Por exemplo: espera que o utilizador diga que esta pronto antes de iniciar o jogo; limpa o terminal após cada jogo; entre outros.

2.1. Descrição das Estruturas de Dados

Para armazenar os diferentes tipos de dados necessários à execução do programa, foram utilizadas diversas estruturas de dados:

- dados: um *array* de inteiros com 3 dimensões que grava todos os dados de todos os jogos por jogador. Isto é, cada jogador tem uma matriz de tamanho 5*3 que guarda se o jogador conseguiu ganhar o jogo, quanto tempo durou o jogo e quantas jogadas foram efetuadas. Consideramos esta como sendo a maneira mais fácil de guardar e aceder a estas informações.
- nome_jogadores: um *array* de caracteres com 2 dimensões, guarda os nomes dos jogadores em forma de lista de *strings*.
- chave: vetor de caracteres (*array* com 1 dimensão) que guarda a chave gerada pelo computador numa *string*.
- copia_chave: vetor de caracteres (*array* com 1 dimensão) que guarda uma copia da chave gerada pelo computador numa *string* para poder modificá-la nas comparações com a tentativa.
- jogada: vetor de caracteres (*array* com 1 dimensão) que guarda a chave introduzida pelo jogador numa *string*.
- copia_jogada: vetor de caracteres (*array* com 1 dimensão) que guarda uma copia da jogada introduzida pelo jogador numa "string" para poder modificá-la nas comparações com a tentativa.
- mediaTempos: vetor de *floats* que guarda a media do tempo de jogo por jogador.

Para além destas estruturas de dados, utilizamos mais variáveis afim de, por um lado, guardar e processar os inputs do utilizador (nomeadamente as configurações) e, por outro, auxiliarem na correta execução do programa (como variáveis ligadas ao tempo).

2.2. Descrição das Funções Principais

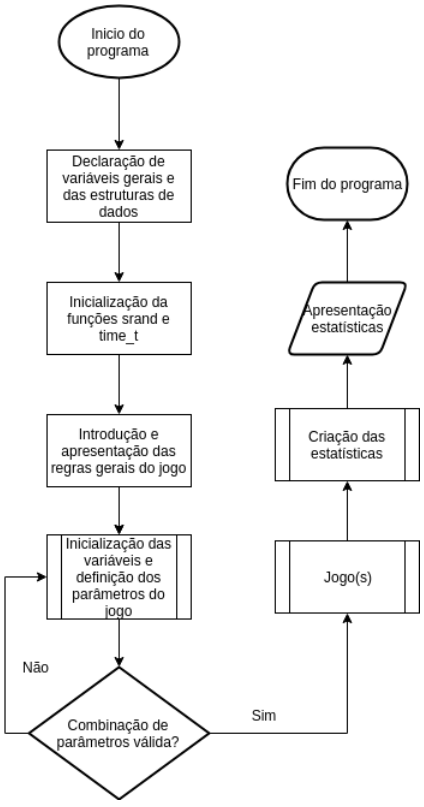
Com o objetivo de deixar a nossa função *main* o mais curta, simples e clara possível, recorremos á utilização de múltiplas funções de seguida descritas:

- introducao: função que dá as boas-vindas ao(s) utilizador(es) e que imprime as regras fundamentais do jogo.
- cleanslate: função utilizada para absorver qualquer *input* extra ou inesperado do utilizador de forma a evitar o vazamento deste para a próxima *input query*, ou a geração de erros de execução.
- clearScreen: função que limpa o terminal após o utilizador clicar no *enter*.
- initialization: tanto esta função como as suas variantes (initializationNames e initializationRepetitions) são utilizadas para inicializar e definir os parâmetros do jogo.

- checkCombinação: função que verifica se a combinação dos parâmetros tamanho_chave, numero_cores e repeticao_cores é válida.
- createKey: função que gera uma chave secreta e aleatória segundo os parâmetros definidos durante as inicializações.
- userAttempt: funcao para confirmar a validade da jogada efetuada pelo jogador. Ou seja, verifica que o tempo não acabou, e que a chave tem o tamanho certo.
- checkInput: função que verifica que os inputs dos jogadores durante o decorrer do jogo são válidos (i.e., são jogadas possíveis). Esta é utilizada dentro da função userAttempt.
- comparaChave: funcao para comparar a chave de jogo com a jogada feita pelo jogador e imprime o número de pinos brancos e pretos.
- jogo: função realiza o torneio (todos os jogos de todos os jogadores), chamando diversas funções: createKey, userAttempt e comparaChave.
- criaDados: funcao que gera os dados de jogo gerais de cada jogador como a media de tempo de jogo e o número de vitorias.
- vencedor: função que determina o vencedor de acordo com os dados dos jogos do torneio. A sua variante (resultados) determina os vencedores nas duas outras categorias.
- ShowData: função que apresenta a performance de cada jogador caso seja pedido

2.3. Fluxograma Geral

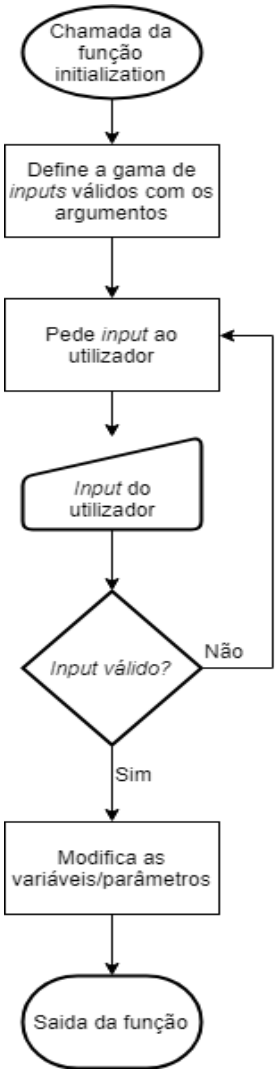
Fluxograma geral do programa:



2.4. **Fluxograma das Funções Principais**

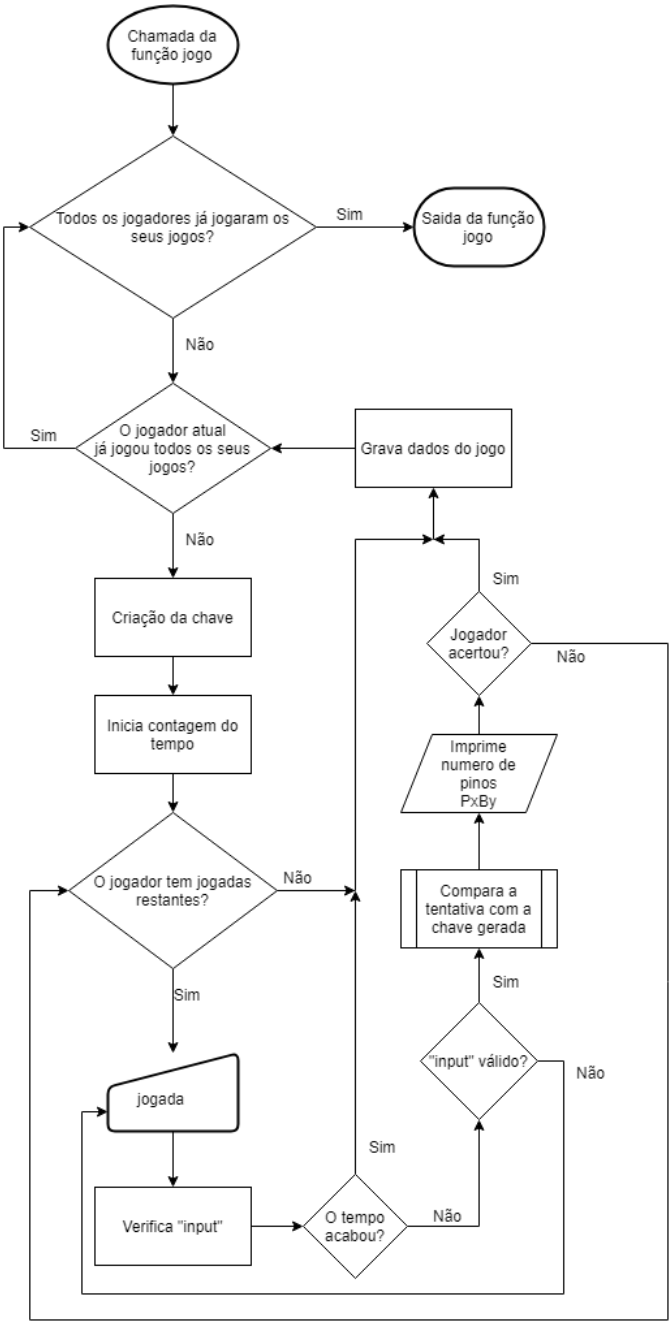
<u>Fluxograma da função “initialization” (e variantes) á</u> esquerda

←



<u>Fluxograma da função “jogo” á direita:</u>

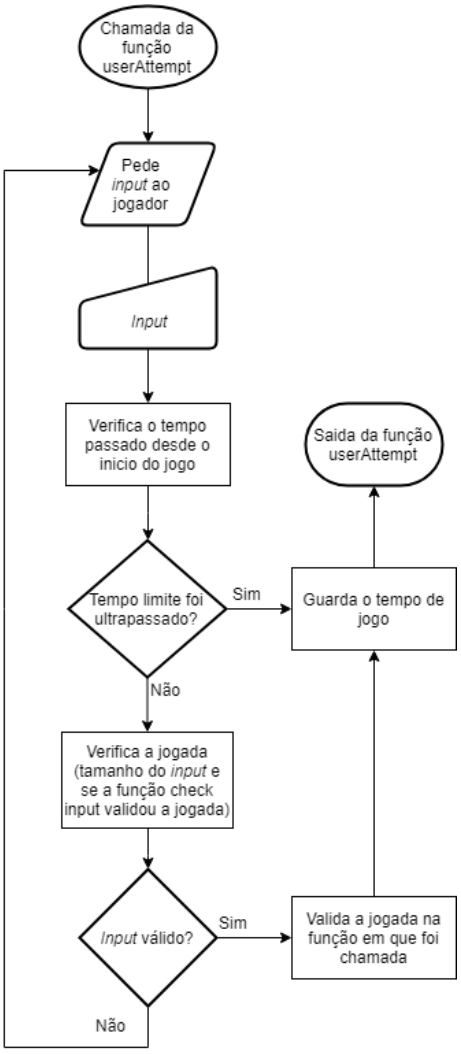
→



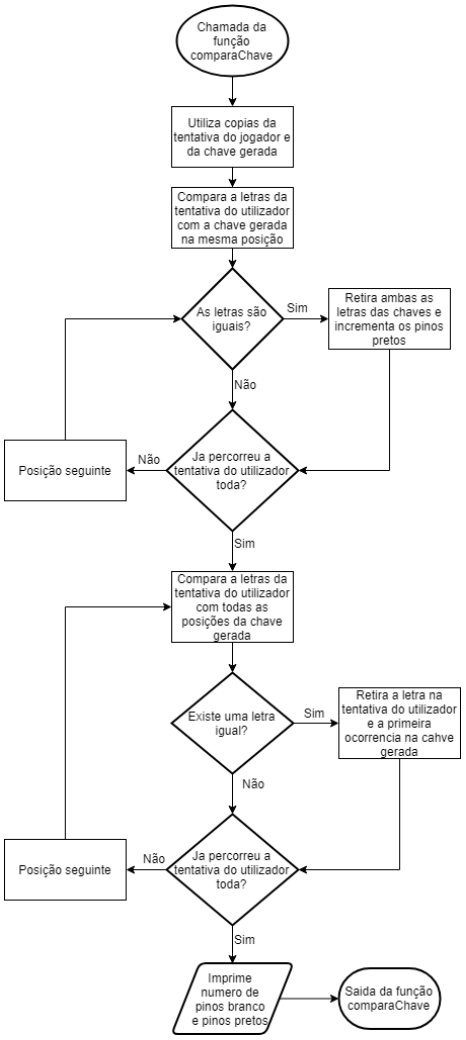
<u>Fluxograma da função “createKey” á esquerda:</u>

←

Fluxograma da função "userAttempt" á esquerda ←



Fluxograma da função "comparaChave" á direita: →



3. Conclusão

Em suma, para além de desenvolver um programa que se limita cumprir as exigências do enunciado, esteve sempre presente a vontade de implementar características suplementares que melhorariam a experiência dos utilizadores/jogadores. Por outro lado, um objetivo importante foi deixar o código fácil e confortável de ler ao utilizar nomes de variáveis facilmente identificáveis e ao apostar fortemente no uso comentários todo ao tanto longo do código como antes de cada função. Por outro lado, com o intuito de verificar a inexistência de erros e *warnings* adicionais, compilamos o nosso programa com as seguintes *flags*: *-Wall -Wextra -ansi -pedantic -std=c99;* e adicionalmente, também corremos o programa com a ferramenta de depuração Valgrind (valgrind --leak-check=full) corrigindo todos os erros e *warnings* encontrados.