Relatório PokerISTars 2020

Grupo 75
André Santos, 96152 – andresantos1.alf@gmail.com
Sebastião Fonseca, 96320 – sebastiaomachado01@gmail.com

Estrutura do programa

Foi proposto o desenvolvimento de um programa que simula um jogo de poker com diversos modos de funcionamento. De modo a manter o programa bem estruturado, foi tomada a decisão de separar o código em vários ficheiros:

- poker.c, contém a função “main” e as funções para verificar a validade dos argumentos.
- global.c, ficheiro que contém funções que são utilizadas ao longo de todo o programa. Estas incluem funções que dividem as cartas, verificam combinações, ordenam as cartas pela ordem pretendida, ou desempatam mãos com o mesmo valor.
- global.h, contém as declarações das funções implementadas no ficheiro global.c.
- c.c, ficheiro que trata o modo ”-c” do programa. Contém também uma função que cria uma string de cartas a partir dos argumentos. Recebe apenas como argumentos “argc” e “argv”, utilizando funções presentes no ficheiro “global.c” para realizar as operações.
- c.h, contém as declarações das funções referentes ao modo consola.
- di.c, trata do modo “-di”, contém as implementações únicas a este modo de funcionamento.
- di.h, tem as definições das funções implementadas no ficheiro di.c.
- dx.c, referente ao modo “-dx”. Contém, para além da função “dx”, funções exclusivas a este modo de funcionamento, já que é o único que compara um elevado número de jogadores num ambiente contínuo.
- dx.h, declara as funções exclusivas ao modo “-dx”
- s.c, contém a função “s1”, que realiza as operações referentes ao modo “-s1” e as funções que este utiliza exclusivamente, nomeadamente manipulação de listas, já que esta é a única função que utiliza listas.
- s.h, contém as declarações das funções utilizadas no ficheiro s.c, também como a definição da estrutura a partir da qual são criadas as listas.

Poker.c, função “main”

A função main tem como objetivo identificar se os argumentos introduzidos pelo utilizados através da linha de comandos são validos. Para tar, foi necessário recorrer a funções de manipulação de strings, como “strncmp” e “strcpy”, de modo a identificar o modo pretendido pelo utilizador e os seus argumentos, bem como a validade destes. Após correta identificação de um modo de execução, é realizada uma chamada à função especifica que realiza o que o utilizador pretende.

Primeiramente, é realizada uma verificação ao número de argumentos do programa. Se este for menor que três, imediatamente é devolvido “-1” e o programa termina, já o programa requer pelo menos 3 argumentos para funcionar corretamente.

De seguida, é realizada uma verificação ao segundo argumento, e chamada uma função dependendo deste que realizada a verificação dos restantes argumentos, para confirmar que são validos para realizar a operação pretendida.

Modo “-c”

Se o modo de operação pretendido pelo utilizador for o de linha de comando, a função principal chama a função c.

Esta função começa por identificar o número de cartas introduzidas pelo utilizador, já que a operação a realizar depende deste valor. Independentemente deste número, a função cria uma string com todas as cartas, bem como um vetor que guarda os valores das cartas pela mesma ordem que foram introduzidas na linha de comandos, e uma string que faz o mesmo para os naipes. É também utilizada uma função para ordenar as cartas de acordo com o seu valor, realizando esta reordenação tanto na string com todas as cartas como nos vetores que guardam apenas os naipes e os valores, no caso de haver dois jogadores, isto é feito duas vezes, uma para cada jogador.

No caso das 5 cartas, basta utilizar uma função que verifica a pontuação da mão utilizando os arrays anteriormente obtidos, depois dar print à pontuação para stdout.

Para 7 cartas, é necessário testar todas as combinações possíveis de modo a encontrar a melhor combinação de 5 cartas que se pode formar. Para isso, utiliza-se uma função que gera todas as combinações, e, caso existam duas mãos com a mesma pontuação, chama-se uma função que as desempata. Após obter a melhor mão possível do conjunto, esta é devolvida para stdout com a sua pontuação.

No caso das 9 cartas, o processo é igual ao das 7 cartas, mas agora repetido para os dois jogadores. São obtidas as melhores mãos de ambos os jogadores, calculadas as pontuações e, caso estas sejam iguais, desempatam-se de modo a obter o vencedor. De seguida, escreve-se em stdout as mãos e as pontuações de ambos os jogadores seguidas do jogador vencedor, ou 0 caso haja um empate.

Por fim, no caso das 10 cartas, repete-se o mesmo processo das 5 cartas para ambos os jogadores. Caso tenham a mesma pontuação, é utilizada a função de desempate para obter o vencedor e devolver para stdout o seu número.

Modo “-di”

A função “di” recebe como parâmetros de entrada o nome do ficheiro que contém os baralhos, o nome do ficheiro para onde serão escritos os resultados caso exista, o modo (1, 2, 3 ou 4, respetivamente para -d1, -d2, -d3, -d4) e um bit “write” que determina se os resultados serão devolvidos para um ficheiro ou para a linha de comandos.

A função começa por abrir o ficheiro de entrada e o de saída caso “write” esteja a 1. De seguida, o programa verifica através de um switch/case qual o modo pretendido, prosseguindo então com as operações necessárias.

No modo -d1, o programa lê 50 das 52 cartas do baralho, de modo a criar 10 conjuntos de 5 cartas. Cada conjunto é então processado através de uma função “di_1” que devolve o número de conjuntos testados, valor utilizado posteriormente para estatísticas. Esta função atualiza também os valores do vetor “stats”, que guarda o número de ocorrências de cada combinação. Este vetor será

utilizado para fins estatísticos. Estas operações estão dentro de um ciclo “while” dependente da função “load_deck”, que retornará 1 quando terminar a leitura do ficheiro. Depois de terminar o loop, é chamada a função estatisticas_1, que trata do processamento de dados e da escrita das estatísticas para apena um jogador.

O modo -d2 funciona de forma semelhante ao -d1, sendo que a única diferença é que serão testadas todas as combinações possíveis com as 7 cartas de cada combinação, utilizando a mais alta. Este modo lê 49 das 52 cartas do baralho, criando assim 7 conjuntos.

Os modos -d3 e -d4 funcionam de maneira semelhante no modo em que ambos tratam das combinações para 2 jogadores. No modo -d3, são lidas 45 cartas, distribuídas por 5 conjuntos. Cada um destes conjuntos é testado, criando as melhores combinações possíveis para cada jogador e determinando o vencedor, de maneira muito semelhante ao modo de 9 cartas da função “-c”.
São guardadas as ocorrências de cada combinação num vetor “stats”, também como as vitorias de cada jogador num vetor “vitorias”. Para o processamento de dados, é agora utilizada a função estatisticas_2, que tratas das estatísticas para os modos de dois jogadores.

O modo -d4 funciona de modo idêntico com a exceção de que existe apenas uma combinação possível para cada jogador. Como tal, é apenas necessário verificar a pontuação de cada jogador e determinar o vencedor. O cálculo de estatísticas é também idêntico ao do modo -d3.

Modo “-dx”

A função “dx” trata o modo de funcionamento do mesmo nome. É primeiramente aberto o ficheiro de onde serão lidos os baralhos de cartas, e um ficheiro de escrita caso o utilizador especifique que a escrita deve ser feita para um ficheiro.

Cada jogador é representado por uma estrutura que inclui os pontos acumulados ao longo do jogo, as cartas em mão do jogador, vetores independentes para os naipes e os valores das cartas, número de vezes que o jogador foi a jogo bem como as que deu fold, os pontos da mão atual, e duas flags, uma que indica que o jogador está inativo e uma que indica que o jogador ganhou a ronda.

Após abrir os ficheiros, é iniciado um ciclo que lê todos os baralhos do ficheiro e executa as operações necessárias para cada um. Para tal é utilizada uma função “load_deck”, que carrega as cartas necessárias para um vetor e que devolverá o valor “-1” quando acabar de ler o ficheiro. É então utilizada uma função que distribui as cartas pelos jogadores ativos, calcula a pontuação de cada mão e determina se determinado jogador dá fold. Para verificar se um jogador deve receber cartas é utilizada uma flag que indica que o jogador está inativo. Esta flag é atualizada na mão anterior utilizando uma função própria.

Após distribuição das cartas e determinação dos folds, é utilizada uma função “Check_vencedores” que verifica qual ou quais os jogadores com mãos vencedoras, e distribui 1 ponto dividido por todos os vencedores. Esta função devolve 0 caso nenhum jogador esteja ativo, indicando que esta ronda não conta para efeitos estatísticos. São então atualizadas as flags de cada jogador e o apontador para o ficheiro é movido para o baralho seguinte.

Para obter a percentagem de vitorias de cada combinação, divide-se o número de ocorrências de cada combinação pelo número de rondas em que houve pelo menos um jogador ativo.

De modo a dar print às pontuações dos vários jogadores por ordem, é utilizada uma função que determina a ordem dos jogadores a partir dos pontos de cada um, prosseguindo-se então à escrita das pontuações para a saída escolhida.

Funções globais

-Check_Hand: Esta função recebe um vetor e uma string, ambos de 5 elementos, que representam os valores e os naipes das cartas de uma mão. A função atribui uma pontuação à mão utilizando pequenas funções auxiliares que detetam cada combinação possível.

-Desempate: Esta função é utilizada para desempatar duas mãos com a mesma pontuação, recebendo dois vetores de 5 inteiros que representam os valores das cartas de uma mão, e um inteiro, a pontuação dessas mão. O método de desempate não é igual para todas as combinações, dependendo da pontuação, a função pode chamar outras funções secundárias que ajudam no processo de desempate. No final, a função retorna o número do jogador vencedor ou zero, caso seja empate.

-Check_combinações: Recebe 2 vetores de inteiros e duas strings, uns de 7 elementos e outros de 5. Os arrays de 7 elementos representam uma mão completa de um jogador, e os arrays de 5 elementos guardam a melhor combinação possível realizar com essas cartas. Utilizando vários loops, a função gera todas as 21 combinações de 5 elementos possíveis a partir das 7 cartas, chama a função check_hand para verificar a sua pontuação, caso seja superior à pontuação máxima verificada até ao momento, guarda essa mão nos arrays de 5 elementos, caso a pontuação seja igual, chama a função Desempate para decidir o que fazer. No final, retorna a pontuação da combinação mais forte.

-Check_string: Recebe uma string, um inteiro com o seu comprimento e um inteiro com o número de cartas que contem, percorre a string e verifica se há alguma carta ilegal, caso houver, retorna -1, senão retorna 0.

-Ordenar: recebe uma string de cartas, uma string de naipes, um vetor com os valores das cartas e um inteiro com o número de cartas. A função ordena os três arrays de modo a que as cartas fiquem ordenadas por valor decrescente e naipe alfabeticamente. Isto é feito em três passos. Primeiro, ordena-se o vetor por ordem decrescente, guardando a ordem num vetor auxiliar para reordenar a string “naipes” da mesma maneira, depois, percorre-se o vetor e, quando se encontra duas cartas iguais, compara-se os naipes e ordena-os alfabeticamente. Por fim, reconstrói-se a string “cartas” utilizando os dois arrays já ordenados. Sendo uma função void, não retorna nada.

-load_valcartas: Recebe um vetor, uma string cartas e um inteiro com o número de caracteres na string. A função lê a string e guarda os valores das cartas no vetor que recebeu.

-load_naipes: Recebe uma string “cartas”, uma string e um inteiro com o número de caracteres da “string” cartas. Semelhantemente à função anterior, lê a string cartas e guarda os naipes na string introduzida.

-load_cartas: Recebe uma string “cartas”, uma string com naipes, um vetor com valores de cartas e um inteiro com o número de cartas. A partir do vetor de valores e da string de naipes, constrói a string cartas.

Modo “-s1”

A função que trata o modo de shuffle recorre a listas para realizar a reordenação. É criada uma estrutura para cada carta, que guarda o seu valor e o seu naipe, para além de um apontador para o elemento seguinte da lista. Com isto, é criada uma lista simplesmente ligada com todas as cartas.

É criado também um vetor que guarda a sequência que se pretende utilizar para baralhar as cartas. De modo a não alocar memória desnecessariamente, é utilizada a função “realloc” de modo a alocar memória de acordo com a necessidade de um maior vetor. Sempre que é adicionado um valor ao vetor que guarda a sequência, é incrementada uma variável que será posteriormente utilizada para percorrer o vetor.

Faz-se então a leitura das cartas do ficheiro para uma string, realizando-se então a distribuição das cartas da string para a lista. Esta distribuição é realizada através de uma função que utiliza um ciclo “for” para alocar memória para cada elemento da lista e colocar os valores corretos neste, retornando um apontador para a cabeça da lista.

Após preenchimento das listas, o vetor ocupado pelos valores da sequência é percorrido, verificando os valores um a um, de modo a baralhar as cartas do modo pretendido. Dependendo do valor, encontrado, é chamada uma função que realiza a operação pretendida antes de seguir para o valor seguinte do vetor.

Caso o valor seja “1”, a lista inicial é dividida em 2, colocando o endereço do elemento 27 num ponteiro auxiliar e o apontador “next” do elemento 26 a null. Após isto, é guardado o valor da cabeça da lista num ponteiro auxiliar, que é utilizado em conjunção com o ponteiro utilizado para guardar o endereço do elemento 27 para percorrer a lista, alternando os valores guardados. É retornado então o header da lista.

Caso o valor seja “2”, a operação realizada é muito semelhante à do caso anterior. O baralho é dividido em 2 da mesma forma, mas desta vez é chamada uma função que realiza a inversão da lista correspondente à segunda metade do baralho. Esta função troca a ordem dos apontadores percorrendo a lista e colocando cada elemento a apontar para o elemento anterior. De seguida, é percorrida a lista tal como no primeiro caso, alternando os valores de ambas as listas auxiliares.

Caso o valor seja “3”, é realizada a divisão do baralho como nas operações anteriores, mas desta vez, é guardado também o endereço da segunda metade do baralho num apontador auxiliar. De seguida, esta segunda metade é percorrida e, ao chegar ao último elemento, coloca-se este a apontar para o primeiro elemento da primeira metade do baralho. É então devolvido o apontador auxiliar que guardou o endereço da segunda metade do baralho.

Após a reordenação do baralho, é utilizada uma função que percorre todos os elementos da lista, imprimindo o naipe e valor de cada um para a saída pretendida. De seguida é realizada a libertação da memória alocada para a lista e para o vetor da sequência.

Fluxogramas

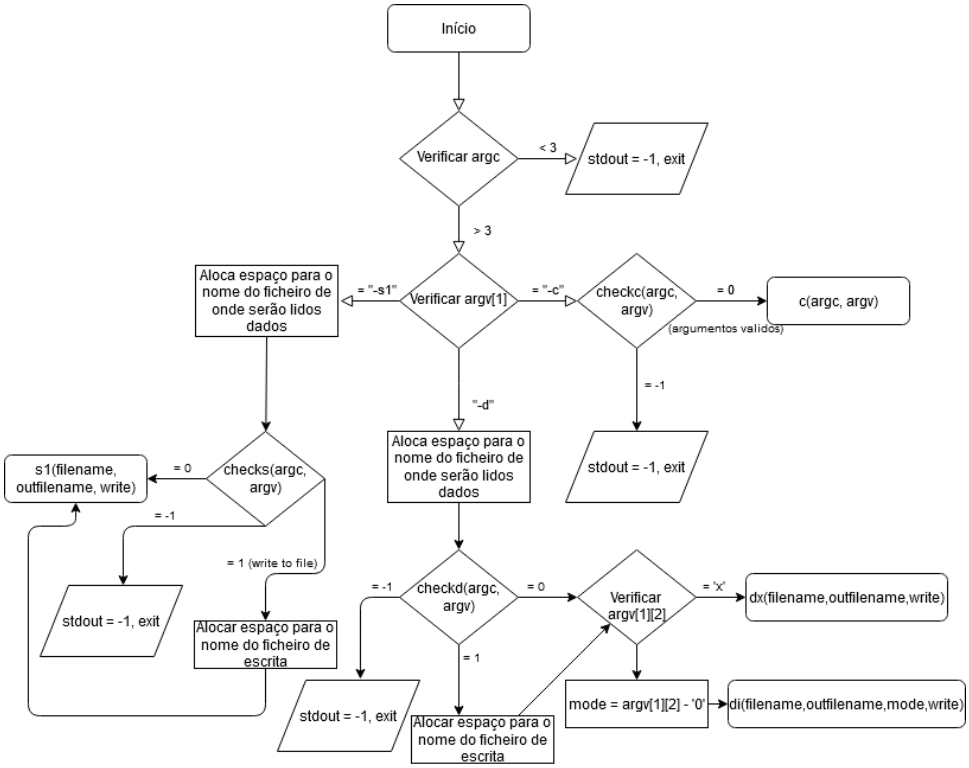

Figura 1. Função "main" - Valida argumentos e prossegue para operações especificas

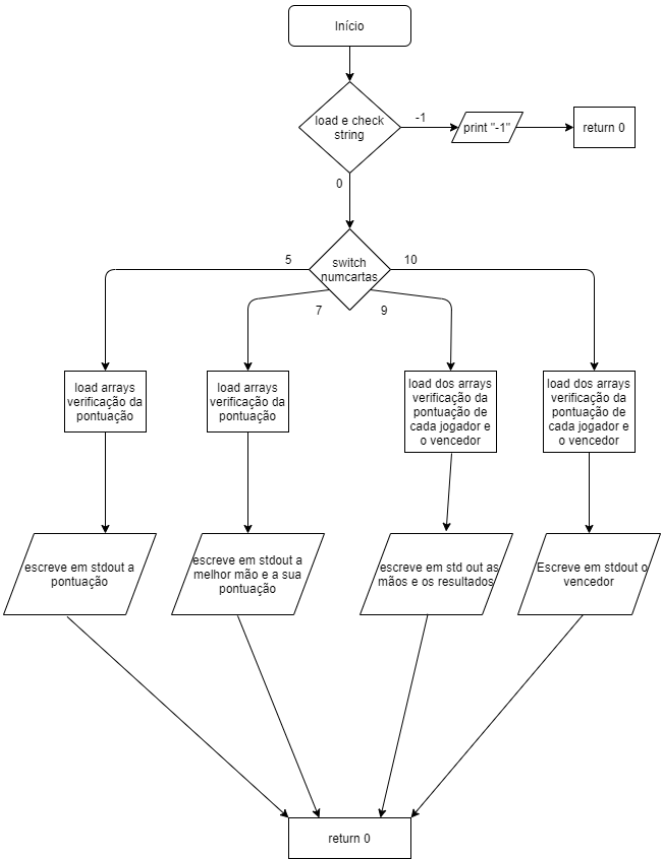

Figura 2. Função "c" - Realiza operações de acordo com o número de cartas introduzidas pelo utilizador.

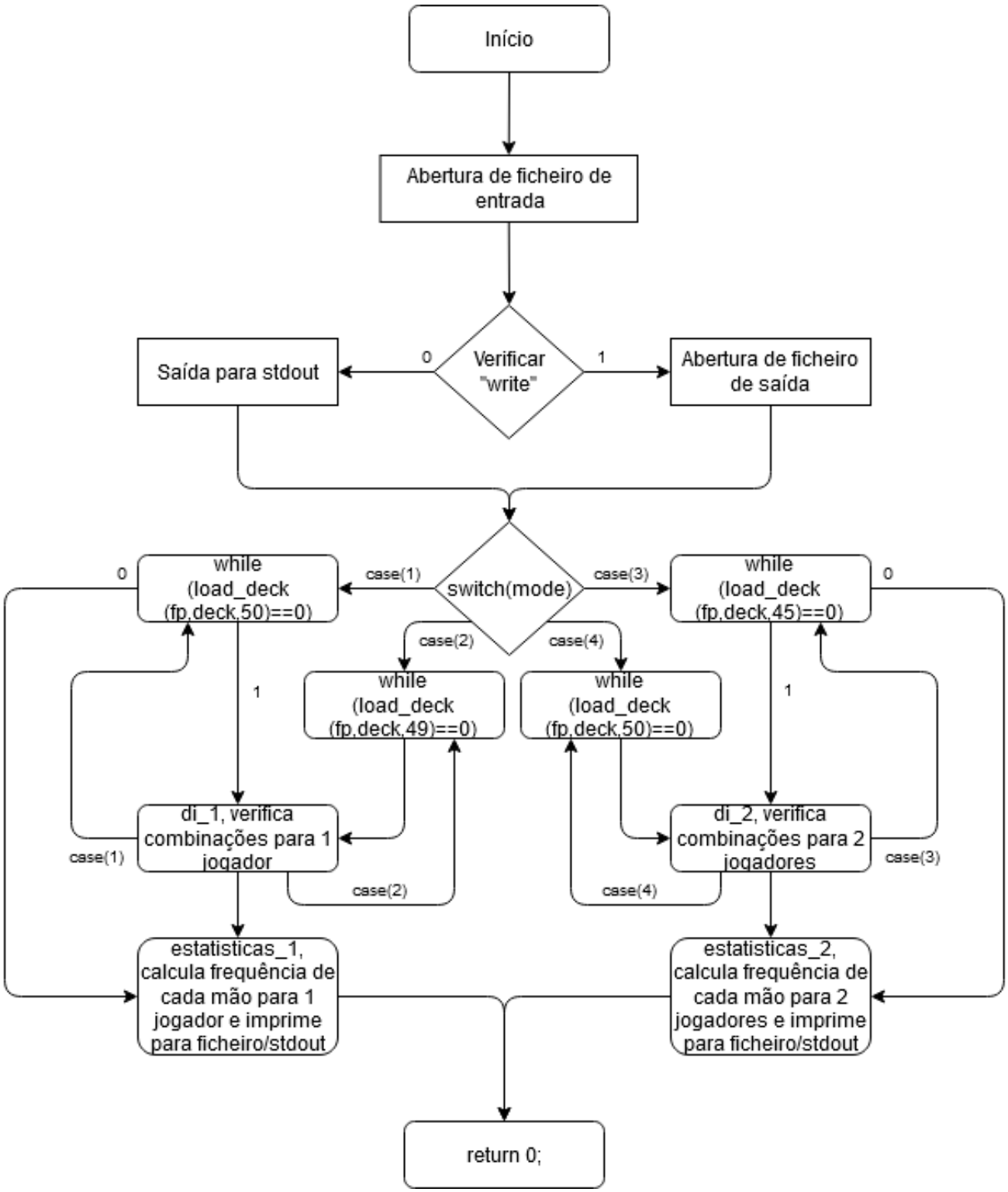

Figura 3. Função "di" - Verifica modo de escrita e modo de processamento, realizando de seguida as operações pretendidas.

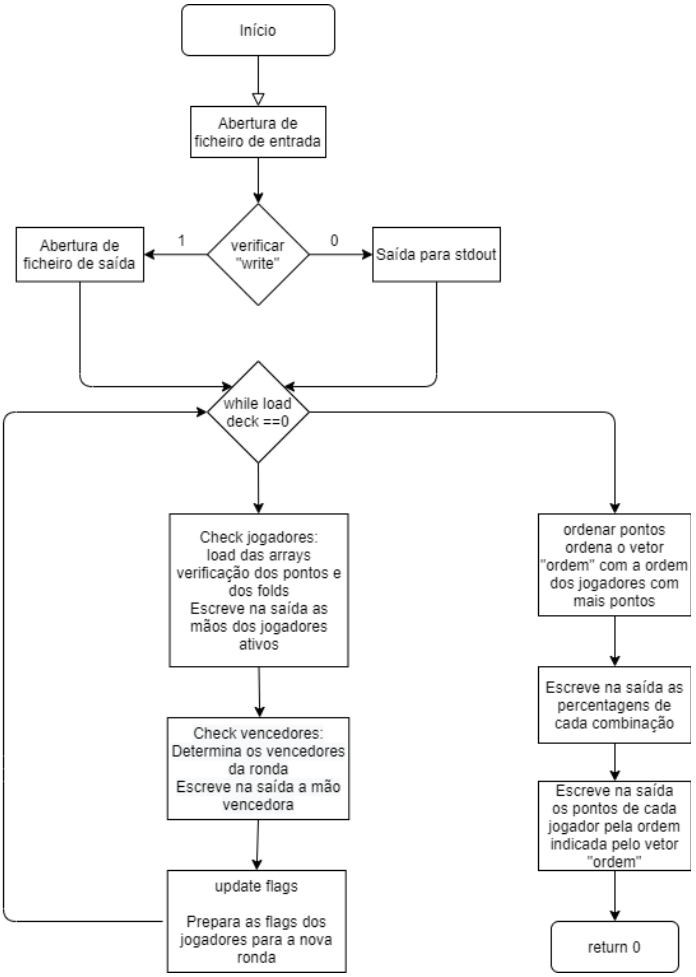

Figura 4. Função "dx" - Processa baralhos um a um realizando as operações relativas à realização do "torneio"

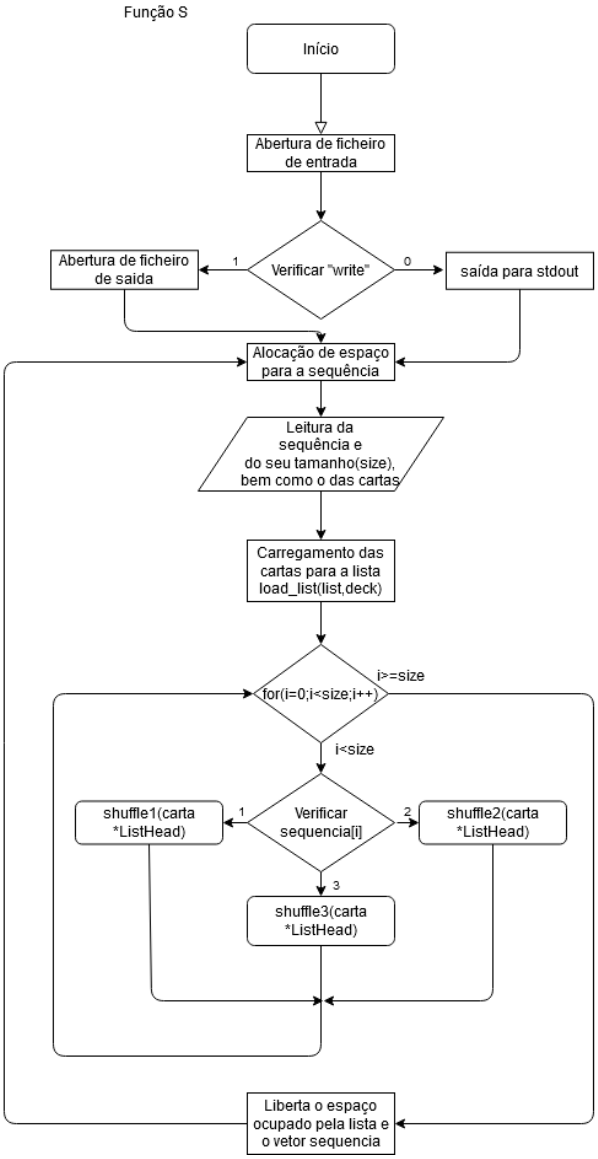

Figura 5. Função "s1" - Verifica modo de escrita e baralha as cartas de acordo com as combinações indicadas no ficheiro.