UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
SSC0600 – INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO
TRABALHO 05 – Qwirkle

Aluno 01: Vitor Favrin Carrera Miguel NUSP: 11800646
Aluno 02: Leonardo Rodrigues de Sousa NUSP:10716380

## Descrição

O programa inicia perguntando qual vai ser o número de jogadores, com essa informação pergunta-se quais são os nomes dos jogadores e logo em seguida pergunta-se se será utilizado o cheat mode ou não. Definidas as informações, cada jogador receberá 6 peças de forma aleatória, as peças e a pontuação de cada jogador são printados na tela.

A partir disso, sempre será indicado qual é o jogador da rodada atual, sendo que as rodadas ocorrem na mesma ordem em que os nomes foram inseridos. Em cada primeira jogada de um jogador são dadas as opções de trocar, jogar e passar. Nas seguintes jogadas do mesmo jogador são dadas apenas as opções de jogar e passar.

Para realizar a opção trocar, o jogador deve digitar “trocar” ou “t” seguido das peças que ele deseja trocar. Assim, procura-se as peças digitadas dentre o total de peças do jogador e, ao encontrar, a peça é retirada e outra é colocada no lugar de forma aleatória. Ao fim desse comando, será a vez do próximo jogador. Para fazer uma jogada, o jogador deve digitar “jogar” ou “j” , seguido da peça que ele deseja jogar e qual será a posição no tabuleiro, respectivamente. Para passar a jogada para o próximo jogador, o jogador deve digitar “passar” ou “p”.

Se o cheat mode estiver desligado, os jogadores só poderão fazer as jogadas com as peças que possuem, caso tentem fazer uma jogada com uma peça que não possuem, será printado uma mensagem de erro e o jogador deverá informar novamente o que deseja fazer. Caso o cheat mode estiver ligado, o jogador poderá fazer uma jogada com as peças que possui, ou com as peças que ainda estão disponíveis.

Observação: No primeiro movimento de todos, o tabuleiro não será apresentado, pois apenas a posição 0x0 será possível para uma jogada. Conforme as jogadas são feitas, é printado na tela o tabuleiro com as jogadas anteriores e quais são os índices que o usuário precisa digitar para colocar uma peça no tabuleiro.

O jogo só acaba quando todas as peças são colocadas no tabuleiro e quando isso acontecer, será informado qual foi a classificação dos jogadores, de acordo com as respectivas pontuações.

## Implementação do código

Os arquivos que contém o código estão divididos em 3. No arquivo main, o jogo é iniciado, toda a memória necessária é alocada, a seed para a geração de números aleatórios é setada, são declaradas variáveis dos tipos das structs declaradas no arquivo “funcoes.h” e são feitas as perguntas sobre: número de jogadores, nomes e se será utilizado o cheat mode. O restante das funções estão divididas em dois tipos: as principais e as auxiliares. As funções principais estão dentro do arquivo “principais.c”, sendo estas responsáveis por receber e interpretar o que o usuário deseja fazer. As funções auxiliares estão dentro do arquivo “auxiliares.c”, sendo estas responsáveis por: checar se o movimento realizado foi válido, atualizar a contagem de pontos, recolocar as peças dos usuários, administrar o total de peças disponíveis, imprimir o tabuleiro, refazer o tabuleiro quando for necessário, checar se o jogo já acabou e desalocar a memória utilizada.

Além dos arquivos citados acima, há também o arquivo “funcoes.h” o qual contém a declaração das structs “jogo” e “contadores” e a declaração das funções principais e auxiliares.

Observação: Para indicar que uma posição não possui peças, é utilizado a string “v0” no lugar.

Descrição dos campos das structs

Com o intuito de simplificar a passagem de parâmetros para as funções, duas structs foram utilizadas no código, a “jogo” e a “contadores”. A struct “jogo” contêm campos relacionados as informações que precisam ser armazenadas durante o decorrer do jogo, os campos são:

int n_jogadores - Número de jogadores

char **jogadores - Nome dos jogadores

int cheat - Flag para cheat mode

int **pontos - Guarda a pontuação total e a pontuação da jogada atual de cada jogador. Ele é de tamanho “n_jogadores” x 3, sendo que para cada jogador o índice 0 armazena a quantidade total de pontos, o índice 1 armazena a quantidade de pontos feitos na linha de uma jogada e o índice 2 será a pontuação feita em uma coluna.

int **total_pecas - Guarda qual é a quantidade de peças disponíveis de cada tipo, sendo que cada posição corresponde a uma peça. É 6x6.

char ***pecas_jogadores - Guarda quais são as peças de cada jogador. É “n_jogadores”x6x2.

int *direcao - Armazena qual é a direção a ser mantida na jogada (linha/coluna), começa com cada valor igual a -1 e muda conforme a direção das jogadas é definida.

int *tam_tabuleiro - Guarda qual é o tamanho do tabuleiro atual

int *deslocamento_x - Determina o deslocamento das peças no tabuleiro em relação as colunas.

int *deslocamento_y - Determina o deslocamento das peças no tabuleiro em relação as linhas.

Descrição dos campos da struct “contadores” :

int flag_primeira_rodada - Serve para dizer se esse é o primeiro movimento de todos.

int n_jogadas - Indica o número de movimentos que foram feitos em uma rodada.

int jog_atual - Índice do jogador atual em “char **jogadores”.

Descrição da funções principais:

int rodada(jogo informacoes, contadores indices, char ***tabuleiro) – Printa quais são os jogadores com suas respectivas peças e pontuação, solicita para o usuário qual deverá ser o comando a ser realizado e chama a função responsável pelo comando.

int passar(jogo informacoes, contadores indices, char ***tabuleiro) - Mostra a pontuação feita na jogada, atualiza a pontuação total do jogador e chama a função “rodada” para o próximo jogador.

int jogar(jogo informacoes, contadores indices, char comando[10][50], char ***tabuleiro) - Recebe qual a peça e a posição que o jogador deseja fazer a rodada, define qual será a direção que deverá ser mantida nas próximas jogadas do mesmo jogador, determina se o jogador possui a peça indicada, checa se o jogador pode utilizar aquela peça para uma jogada. Caso a jogada tenha sido validada, o tabuleiro é preenchido e chama-se a função “imprime_tabuleiro”, caso contrário, uma mensagem de erro é impressa e chama-se a função “rodada” para o jogador refazer a jogada.

int trocar(jogo informacoes, contadores indices, char comando[10][50], char ***tabuleiro) – Recebe quais são as peças que o jogador deseja trocar por meio do parâmetro “comando” e procura estas peças dentre as peças que ele possui. Caso ache a peça solicitada, a peça é removida da mão do jogador e devolvida ao total de peças por meio da função “total_pecas” no modo 1, depois a função “poe_pecas” é chamada para repor a mão do jogador com peças aleatórias.

Descrição das funções auxiliares:

int checa_mov(jogo informacoes, int linha, int coluna, char *jogada, int jog_atual, char ***tabuleiro) – Depois que a função “jogar” valida que determinada peça pode ser utilizada para uma jogada, a função “checa_mov” checa se a peça pode ser colocada no conjunto do tabuleiro, retornando 1 caso seja possível e 0 caso contrário. A posição e a peça são passados para a função por meio dos parâmetros “linha”, “coluna” e “jogada”. A partir disso, conta-se qual é a quantidade de letras e números iguais ao que foi passado e a informação é guardada dentro das variáveis “cont_letra” e “cont_numero”, respectivamente. Desse modo, sabe-se que se, em determinada direção “cont_letra” for igual a “cont_numero” ou que uma das duas for diferente de zero e diferente do total de peças daquela direção, o movimento é inválido e então retorna-se zero. Com os valores adquiridos, é possível determinar a pontuação feita até o momento, mas para isso utiliza-se a função “atualiza_pontos”.

Observação: Há um detalhe diferente do vídeo. Durante a apresentação, o trecho de código que fazia a adição da última peça ao total de peças estava no final da função “checa_mov”, para manter a coerência com o propósito de cada função, este trecho foi movido para o inicio da função “atualiza_pontos”.

void atualiza_pontos(jogo informacoes, int total_linha, int total_coluna, int jog_atual) – Os parâmetros “total_linha” e “total_coluna” correspondem a pontuação que foi feita na linha e na coluna, respectivamente. Como uma das duas direções é mantida durante as jogadas de um jogador, significa que nesta direção as peças já haviam sido contadas antes, então basta substituir a pontuação antiga pela nova, enquanto que a outra direção deve ser acumulada ao total.

void poe_pecas(jogo informacoes, int jog_atual) – A função recebe qual é o jogador que deve ter as peças recolocadas por meio do parâmetro “jog_atual”, passa por todas as peças que ele possui e quando encontrar “v0” sabe que deve substituir por alguma peça. Se as peças em “total_pecas” já tiverem acabado nada é feito.

int imprime_tabuleiro(jogo informacoes, contadores indices, char ***tabuleiro) – A função começa definindo até onde as peças vão dentro do tabuleiro e seus valores são gravados nas variáveis linha_superior, linha_inferior, coluna_esquerda e coluna_direita. Caso as peças

estejam chegando perto da borda do tabuleiro, a função “novo_tabuleiro” é chamada. Depois o tabuleiro é printado de forma que os índices sejam indicados aos usuários.

int novo_tabuleiro(jogo informacoes, int linha_superior, int linha_inferior, int coluna_esquerda, int coluna_direita, contadores indices, char ***tabuleiro) – Essa função só é chamada pela função “imprime_tabuleiro” e apenas quando for necessário mudar a disposição das peças no tabuleiro. O primeiro passo é calcular qual é a média das distâncias das bordas, pois caso ainda tenha espaço no tabuleiro as peças serão reposicionadas de forma que fiquem o mais longe possível das bordas. Além disso, uma cópia das peças do tabuleiro é feita dentro da matriz de char “copia”. Caso não tenha como reposicionar as peças, o tamanho do tabuleiro é aumentado por meio do calloc e as nova disposição das peças no tabuleiro é calculada. De qualquer forma, as peças são recolocadas no tabuleiro respeitando a nova disposição.

int total_pecas(char letra, char numero, jogo informacoes, int modo) - A função total_pecas() serve para adicionar ou retirar peças diretamente da matriz “total_peças”. Ela percorre toda a “total_peças” até encontrar a posição equivalente da letra e do número passados pelos parâmetros. Se estiver no modo de adicionar peças ela incrementa 1 nessa posição, se estiver no modo de retirar peças ela decrementa 1. Além disso, caso ela tenha conseguido realizar a operação ela retorna 1, caso contrário 0.

int fim_jogo(jogo informacoes, char ***tabuleiro) - A função “fim_jogo” checa se o jogo já acabou, o jogo acaba quando não há mais peças disponíveis. Para checar isso, a função percorre a matriz “total_peças” e checa os valores guardados, se todos os valores forem 0 isso significa que não há mas peças disponíveis e ela retorna 0, caso contrario ela retorna 1.

void desaloca(jogo informacoes, char ***tabuleiro) - essa função desaloca, um por um, todos os espaços de memória alocados durante a execução do programa e chama a função exit(0).

Link para vídeos:

Descrição do código
https://drive.google.com/file/d/1mrwsHSrOkfnXnQKIiSaOAkkAOwW7eOHQ/view?usp=sharing

Exemplo de funcionamento
https://drive.google.com/file/d/1bgNHPvRRm7OjCbf6LdDPpr9KzRR7nJlD/view?usp=sharing