UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

Raul Cotrim Ferreira - 10748330

Relatório - Trabalho 05

São Carlos/SP
2020

1. Funcionamento do Programa

Ao iniciar o programa, é solicitado ao usuário o número de jogadores. Após a determinação do número de jogadores, é passado por cada um dos jogadores solicitando o nome do jogador. Por fim, é perguntado ao usuário se o jogador quer utilizar o cheat mode.

Após a determinação das configurações dos jogadores e do jogo, o jogo é iniciado. No jogo, apresenta-se um menu com informações e opções a cada jogada do jogador. A cada jogada, é apresentado ao usuário o tabuleiro e as informações de todos os jogadores, sendo elas a pontuação e as peças, assim como apresenta qual o jogador que está realizando a jogada e as suas peças. Além disso, o usuário pode selecionar uma das seguintes opções:

- ➢ Trocar alguma(s) peça(s), indicando quais peças gostaria de trocar. Após essa escolha, sua jogada é finalizada e é iniciada a jogada do jogador seguinte.
- ➢ Jogar alguma peça, indicando a peça e sua posição (coluna x e linha y). Após essa escolha, apresenta-se um segundo menu explicado abaixo.
- ➢ Passar a vez, finalizando sua jogada e passando a vez para o próximo jogador.

Se o jogador selecionar a opção de jogar uma peça, é apresentado para o mesmo um segundo menu, o qual apresenta o tabuleiro e as informações dos jogadores (semelhantemente ao menu acima) e o usuário pode selecionar uma das seguintes opções:

- ➢ Jogar alguma peça, da mesma forma que no primeiro menu. Se o jogador ainda apresentar peças em sua mão, o jogo apresenta novamente o segundo menu.
- ➢ Passar a vez, finalizando sua jogada e passando a vez para o próximo jogador.

Após a finalização da jogada do jogador, é apresentado a pontuação recebida pelo jogador na jogada realizada, assim como a pontuação total do mesmo. Após isso, é apresentado o primeiro menu para o próximo jogador.

O jogo permanece nesse ritmo até que acabem-se as peças presentes no monte de troca e todos os jogadores passem sua vez ou acabem com todas as peças de suas mãos. Após a finalização do jogo, é apresentado o vencedor ou os vencedores, em caso de um empate entre dois ou mais jogadores.

2. Condutas de um Bom Usuário

O projeto apresenta verificações de erros nas entradas do usuário:

- ➢ Como apresentado nas regras do jogo, o jogo QWIRKLE é jogador por dois a quatro jogadores.
- ➢ O nome dos jogadores devem ser iniciados por letras (maiúsculas ou minúsculas) . A apresentação de caracteres especiais, assim como espaços é permitida. Entretanto, o nome de jogadores deve ter no máximo 18 caracteres.
- ➢ A escolha do *Cheat Mode* deve ser feita pela impressão de ‘S’ ou ‘N’.
- ➢ A troca de peça(s) deve ser feita pelo comando ‘trocar ’ seguido pelas peças a serem trocadas, separadas por espaços.
- ➢ A jogada deve ser feita pelo comando ‘jogar ‘ seguido pela peça e sua posição (x e y) separado por espaços.
- ➢ Para passar a jogada, deve ser feito o comando ‘passar’.

3. Comentar Funções e Arquivos

O programa apresenta muitas funções grandes e de dificil explicação sem a possibilidade da apresentação do código (uma vez que existem funções que não são possíveis de serem apresentadas em forma de imagem). Dessa forma, será explicado a funcionalidade de cada função separadamente. Além disso, uma vez que existem funções de menor relevância para a solução, será apresentado a explicação para as funções de mais importância, enquanto as funções menos importantes serão tratadas internamente à explicação dos arquivos .c.

- Função “verNumPecas” (jogo.c):

A função “verNumPecas” atua na contagem de peças que um jogador apresenta. Ao percorrer as peças do jogador e compará-las com “uma peça vazia” (considerada no programa como uma string vazia), a função retorna o número de peças que o usuário possui na mão.

- Função “definirJogadores” (jogo.c):

A função “definirJogadores” atua na inicialização dos dados do jogo. Inicialmente cria uma struct “infoJogo” e determina o número de peças para cada peça presente no jogo. Posteriormente, solicita o número de usuários, permanecendo nesse paço até que o usuário entre com um valor inteiro entre 2 e 4. Após a determinação dos números de jogadores, inicia um laço com esse número a fim de determinar o nome do jogador, introduzido pelo usuário (o qual é verificado se apresenta o máximo de 18 caracteres e é iniciado por uma letra), alem de salvar na struct “infoJogador” dados pertinentes sobre o jogadores, como suas peças e pontuação inicial. Por fim, solicita ao usuário a escolha do *Cheat Mode*, permanecendo na solicitação enquanto o usuário inserir algo diferente de ‘S’ e ‘N’. Após a determinação desse modo, a função retorna a struct “infoJogo”.

- Função “reajeitarPecas” (jogo.c):

A função “reajeitarPecas” atua na reposição de peças dos jogadores. Dentro da função, é percorrido as peças do jogador e, caso apresente peças de reposição disponíveis no jogo, seleciona uma peça possível aleatoriamente e coloca na mão do jogador. Caso não apresente mais peças de reposição, o laço é finalizado, assim como a função.

- Função “jogarPeca” (jogo.c):

A função “jogarPeca” atua na jogada da peça, verificando, inicialmente, se a peça em questão apresenta-se na mão do jogador. Após a verificação, verifica-se se a posição x e y entrada pelo usuário são possíveis de acordo com as peças anteriores (verificando se x ou y é mantido em todas as peças jogadas pelo usuário. Caso seja a primeira peça a ser jogada, são salvas as posições da peça a serem comparadas com as próximas peças da jogada. Caso a posição seja possível, a posição é verificada no tabuleiro através da função “posicaoAceita”. Caso a posição seja aceita, a matriz é realocada, através da função “realocarMatriz”, e a peça introduzida na posição, pela função “colocarMatriz”, retirando a peça da mão do jogador caso o mesmo apresente a peça (não apresentaria se estivesse no *Cheat Mode*).

- Função “trocarPecas” (jogo.c):

A função “trocarPecas” atua trocando as peças do jogador. Recebendo as posições das peças na mão do jogador a serem trocadas, a função simplesmente altera as peças presente nessas posições por peças aleatórias da pilha de reposição.

- Função “menuJogada” (jogo.c):

A função “menuJogada” é a função principal da jogada. Inicialmente, entra em um laço para o primeiro menu (apresentado no funcionamento do programa), apresentando o tabuleiro, as informações dos jogadores e o menu de escolha da jogada, pelas funções “printTabuleiro”, “printInfoJogadores” e “printJogada”, respectivamente. Após a entrada da escolha do jogador, inicialmente é feita uma verificação acerca da primeira letra da string inserida, pois, caso a primeira letra não seja ‘p’, ‘t’ ou ‘j’, a escolha já está inválida. Caso essa verificação seja satisfeita, a string é percorrida a cada espaço (‘ ‘) encontrado, sendo, na primeira porção da string, comparado a porção da string com as escolhas “trocar”, “jogar” e “passar”. Caso for verificado que a porção é “trocar”, as poções seguintes da string são

comparadas com as peças do jogador, salvando as posições das peças a serem trocadas. Para o caso de uma peça não presente na mão do jogador, a jogada é iniciada novamente, mas, caso todas as peças estejam presentes, a função “trocarPecas” é chamada e a função é finalizada. Caso for verificada que a porção é “jogar”, a porção seguinte é comparada com as peças da mão do jogador e as seguintes, salvas como as posições x e y. Caso a peça não esteja presente, assim como as posições x e y forem inválidas, ou for apresentado mais coisas na string após a determinação de y, a jogada é iniciada novamente. Porém, caso a peça serja possível e x e y salvos, a função “jogarPeca” é chamada e uma flag é determinada para a entrada em um segundo laço para o segundo menu (apresentado no funcionamento do programa). Por fim, caso for verificado que a porção é “passar”, verifica-se se existe algo a mais na string. Caso haja alguma coisa extra, a jogada é iniciada novamente. Entretanto, caso seja verificado que não há nada além de “passar”, a função é finalizada. Caso tenha sido jogado um a peça, será iniciado um laço para o segundo menu. No segundo menu, são realizadas as mesmas verificações apresentadas no primeiro menu, sendo diferente somente na não possibilidade de troca de peças e em algumas diferenças se a opção “passar” for escolhida, mas apresentando as mesmas verificações para as funções de “jogar”. Para esse menu, caso a função “passar” seja selecionada, a função determina a pontuação da jogada (através da função “pontuar”), apresenta a pontuação do jogador (da jogada finalizada e total), assim como reajeitar as peças do jogador por meio da função “reajeitarPecas”. Vale ressaltar que caso outra peça seja jogada, a função mantém-se no laço até que seja escolhido a opção de “passar” ou que não existam mais peças na mão do jogador.

- Função “limpar” (jogo.c):

A função “limpar” atua na liberação de memória utilizada. Na função, são liberados os jogadores e a matriz do tabuleiro.

- Função “verVencedor” (jogo.c):

A função “verVencedor” verifica o(s) vencedor(es). Na função é percorrido os jogadores, de modo a salvar as posições dos jogadores que apresentam a maior pontuação. Caso exista somente um jogador com tal pontuação, o vencedor é apresentado pela função “printVencedor”, enquanto para um número maior que um de jogadores, o empate é apresentado pela função “printVencedores”.

- Função “menu” (jogo.c):

A função “menu” apresenta-se como a função principal do código. Inicialmente, são iniciados todos os dados de importância para o funcionamento do programa, como os jogadores, informações do jogo e o tabuleiro, através das funções “definirJogadores” e “iniciarTabuleiro”. Após essa inicialização, é iniciado o laço que o programa permanece. Nesse laço, são percorridos os jogadores e, se o número de peças do jogador (verificada na função “verNumPecas”) for maior que zero, é apresentado o menu de jogada do jogador “menuJogada”. Caso contrário, a vez do jogador é passada. Ainda dentro do laço, o programa verifica quantos jogadores seguidos passaram a vez, quebrando o laço uma vez que o número de peças possíveis de reposição e troca for zero e todos os jogadores tiverem passado sua vez. Por fim, é apresentado o(s) vencedor(es), através da função “verVencedor”, e liberado a memória utilizado no programa, através da função “limpar”.

- Função “menuJogada”:

A função “menu”, presente no arquivo “jogo.c” (e consequentemente em seu header “jogo.h”), é a função base do programa. Inicialmente, a função chama a função “definirJogadores” (presente no mesmo arquivo), que será explicada abaixo, e determina as características de fim do jogo falsa, assim como determina o número total de peças possíveis de serem trocar/compradas, assim como inicia o tabuleiro pela função “iniciarTabuleiro” (presente no arquivo “tabuleiro.c”). Após isso, é iniciado o laço que o programa fica preso enquanto a variável “fim” presente na

struct “jogo” permanecer FALSE (0). Dentro desse laço, cada iteração a variável “i” é incrementada, alterando o jogador da jogada. Com o jogador é chamado a função “verNumPecas” (presente no mesmo arquivo) que conta quantas peças o jogador apresenta, chamando a função “menuJogada” (presente no mesmo arquivo), a qual será explicada abaixo, se o número de peças do jogador for maior que 0. Caso contrário, a vez do jogador é passada. Após a jogada , a função verifica o número de jogadores seguidos que passaram a vez. Uma vez que o número de peças de compras do jogo chega a zero, se o número de jogadores que passaram a vez for igual ao número de jogadores, ou seja, todos os jogadores passaram a vez, a variável “fim” da struct é atribuída com TRUE (1). Dessa forma, o laço é quebrado e são chamadas a funções “verVencedor” (presente no mesmo arquivo), a fim de determinar o(s) vencedor(es), e “limpar” (presente no mesmo arquivo), a fim de liberar toda a memória utilizada durante o programa.

- Função “iniciarTabuleiro” (tabuleiro.c):

A função “iniciarTabuleiro” iniciar a matriz de tabuleiro do jogo. Na função, é iniciado a matriz de strings com uma única linha e coluna, determinando a string nessa posição como uma string vazia. Além disso, são determinados o centro e o máximo e mínimo da matriz (em x e y) como zero.

- Função “pontuar” (tabuleiro.c):

A função “pontuar” tem por finalidade retornar a pontuação gerada pelas peças introduzidas na jogada do usuário. Inicialmente, é verificado se as posições das peças apresentadas foram introduzidas em mesmo x ou y. Com essa informação, é percorrido todas as posições, verificando, para cada posição, as posições de mesma linha e coluna que a posição em questão. Verifica-se, partindo da posição em questão, as posições de mesmo x, mas y maiores e menores que o y da peça a ser verificada. Dessa forma, um somador é incrementado a cada string não vazia encontrada. Vale ressaltar que:

- Caso encontre nenhuma peça para algum dos sentidos de procura (x mais, x menos, y mais e y menos, o somador mantém-se inalterado, uma vez que não houve conexão para pontuar;
- Uma vez que podemos pontuar a mesma sequência de peças, no caso de mais de uma peça presente na mesma linha, a cada iteração, é salvo as peças encontradas por outras pelas em algum dos sentidos de procura, a fim de ignorar tais sentidos quando for pontuar a peça encontrada;
- No caso de uma peça inserida no meio de uma sequência de peças, ou seja, seria pontuado para dois sentidos de mesma direção (x ou y), verificamos e salvamos a pontuação realizada para o lado maior (x mais e y mais) a fim de ser considerada na pontuação para o lado menor (x menos e y menos).

- Função “compPecas” (tabuleiro.c):

A função “compPecas” atua na comparação de uma peça com uma sequência de peças, verificando se a mesma já não se apresenta na sequência ou não apresenta a mesma letra ou número. Na função, é percorrido todas as peças, verificando se a peça em questão apresenta a mesma letra ou número. Caso seja verificado que a peça apresenta os dois (a peça já está presente na sequência) ou nenhum dos dois (não apresenta nem a mesma letra nem o mesmo número), a função retorna zero. Caso contrário, retorna 1.

- Função “posicaoAceita” (tabuleiro.c):

A função “posicaoAceita” verifica se a peça é possível de ser inserida na posição inserida. Inicialmente é verificado se a posição em questão apresenta-se fora dos limites do tabuleiro ou se já apresenta uma peça inserida. Caso contrário, a função contínua verificando se existe a presença de um espaço vazio entre a peça inserida e a primeira peça inserida na jogada (se não for a peça inicial). Após essa

determinação, é verificado se existe alguma peça ao entorno da posição a ser inserida, uma vez que não é possível fazer uma jogada separada de todas as peças (ignorando a primeira jogada do jogo que não apresenta nenhuma peça). Por fim, são verificados todos os sentidos possíveis (x mais, x menos, y mais e y menos) chamando a função “compPecas” para verificar se a peça a ser inserida pode ser inserida com as peças que apresentam-se nessas linhas.

- Função “colocarMatriz” (tabuleiro.c):

A função “colocarMatriz” simplesmente insere a peça na posição x y determinada.

- Função “realocarMatriz” (tabuleiro.c):

A função “realocarMatriz” realoca a matriz, aumentando seu tamanho se necessário. Inicialmente, é verificado se a porção y da posição apresenta-se nos limites da matriz. Caso seja verificado que sim, se for apresentado no limite negativo, o mínimo em y é decrementado e o centro de y incrementado, mas se for no limite positivo, o máximo em y é incrementado. Com isso, a matriz é realocada de acordo com o novo tamanho determinado pelo máximo e mínimo em y, alocando as novas posições para a inserção das peças, além de determinar tais strings como vazias. Além disso, caso a posição tenha-se apresentado no limite inferior, os elementos da matriz são deslocados para a linha seguinte, de modo a permanecer cada elemento fixo na posição relativa x e y. Posteriormente a isso, as mesmas verificações são feitas para a porção x. Semelhantemente, a matriz é realocada com o novo tamanho determinado pelo máximo e mínimo em x atualizados, determinando as novas posições como strings vazias. Além disso, no mesmo caso para y, caso a variável x tenha sido apresentada no limite inferior da matriz em x, os elementos da matriz devem ser deslocados, de forma a manterem suas posições relativas x e y.

4. Link Github

https://github.com/raul00cf/Trabalho5LabICC/

5. Link Vídeo com a apresentação da solução

https://drive.google.com/file/d/1Zg0udjOwrq1gJauH9rT0lJJkGDv1fTMi/view?usp=sharing

6. Link Vídeo com a execução do programa

https://drive.google.com/file/d/1BRQYHV7pzUmVOXFlu7m4rD5F3udKDgSt/view?usp=sharing