UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

RELATÓRIO DO TRABALHO 05 DE INTRODUÇÃO À CIÊNCIA DE COMPUTAÇÃO (QWIRKLE)

Participantes do grupo:

1. **Nome:** Felipe de Alcântara Tomé
   N° USP: 11800970

2. **Nome:** Ana Julia Aguiar Tagliassachi
   N° USP: 11800632

3. **Nome:** Gustavo Romanini Gois Barco
   N° USP: 10749202

1. Estruturação

Foi pensada uma estrutura ramificada para o código, no sentido de que a função main, presente no arquivo “main.c”, se comunica com outras funções para montagem do programa. As demais funções foram agrupadas de acordo com a sua relação com uma parte do funcionamento do programa, sendo o arquivo “tab.c” apenas para funções relacionadas ao tabuleiro como o dimensionamento e impressão dele. O arquivo “entrada.c” comporta a função que configura o jogo (nomeia os jogadores, da a eles as peças iniciais etc.). No arquivo “pecas.c” são inicializadas todas as peças do jogo na função inicializaPecas e randomizadas ao serem colocadas no vetor pecas (referência ao que seria o saco de peças, no jogo real); na função trocaPecas, é feita a troca das peças de acordo com o comando do usuário ao jogar. No arquivo “jogada.c”, por sua vez, é feito o procedimento de checagem para analisar se o jogador pode efetuar a jogada na posição solicitada (função verificaJogada) e a jogada propriamente dita na função jogada, onde é substituída uma parte vazia do tabuleiro por uma peça da mão do jogador.

2. Structs

Para facilitar a lógica do programa, foram usadas algumas structs, declaradas no arquivo “struct.h”. São elas:

- peca, que é composta por uma letra e um número, conforme pedem as regras do jogo;
- jogador, que possui um nome de até 40 caracteres (excetuando o ‘/n’ e o ‘/0’), um vetor de peças (struct peca), de 6 posições e a sua quantidade de pontos.

3. Funções

3.1. trocaPecas

Esta função recebe como parâmetros as peças que o jogador que gostaria de trocar as peças tem em mãos, a letra e o número da peça que será trocada e o “saco” de peças. Ela funciona da seguinte forma: o programa inicia um laço para encontrar a peça solicitada na mão do jogador, após encontrar essa peça ele busca a primeira posição vazia do saco de peças, randomiza uma posição para inserir a peça a ser trocada(esta posição estará preenchida por uma outra peça qualquer) em seguida é feita a transferência da peça que estava nesta posição aleatória para o início do saco e a mão do jogador recebe uma peça do fim do saco.

3.2. inicializaPecas

A função que inicializa peças recebe como parâmetro o vetor que comporta todas as peças. Ela funciona utilizando um vetor de int para guardar as posições já utilizadas do vetor de peças, foi feito isso para evitar overwrite de uma peça em outra. O primeiro passo é randomizar a posição para a peça atual e posicionar a peça na posição randomizada do vetor de peças, após a posição é guardada no vetor para posições já utilizadas, as seguintes peças deverão ter suas posições randomizadas diferentes de todas as posições no vetor de posições usadas, evitando assim a substituição de peças.

3.3. configuraJogo

Essa função configura o jogo: recebe o nome de cada jogador, distribui 6 peças para cada um e inicializa suas pontuações como 0.

O primeiro laço é feito para receber o nome dos jogadores. Cada nome é recebido como uma string de tamanho 50, sendo o tamanho máximo desejado de 40 caracteres. Assim foi feito para ser possível verificar se tal entrada é maior que o tamanho desejado. Para tanto, existe, dentro desse laço maior, um primeiro laço menor que verifica o tamanho da entrada. Caso seja maior que o desejado, o programa alerta ao usuário e pede para reinseri-lo.

Satisfeita essa primeira condição, a função entra em um segundo laço menor que verifica se o nome desejado já foi inserido. Caso tenha sido, novamente alerta o usuário de tal problema e pede para inserir um nome diferente. Sendo o nome de tamanho adequado (menor que 40 caracteres, excetuando o ‘\n’ e o ‘\0’) e não tendo nomes repetidos, os nomes estão atribuídos corretamente.

Nesse ponto, há um segundo laço que atribui 0 à pontuação inicial de todos os jogadores e, por fim, usa um último laço menor que distribui 6 peças para cada jogador, pegando-as do “final” do saco de peças ao “início” (do fim do vetor ao começo).

3.4. imprimeTab

A função imprimeTab imprime o tabuleiro do jogo, que é uma matriz de structs “peca”. Para tanto, recebe como parâmetros o tabuleiro e a quantidade de linhas e colunas a serem impressas. Essas quantidades são variáveis ao longo do jogo e são armazenadas nas variáveis “linha” e “coluna”, sendo elas aumentadas conforme as necessidades do jogo. No primeiro laço, são imprimidos os índices superiores das colunas.

Em seguida, é imprimida linha por linha. Para que o tabuleiro seja imprimido na formatação desejada, de maneira minimamente simétrica, são feitas algumas manipulações para impressão. Para imprimir cada linha, a função imprime seu índice do lado esquerdo, imprime cada posição da linha, estando ela ocupada por uma peça ou não - nesse último caso, a posição é ocupada por espaços - e cada

posição é separada por uma barra. É impresso também o índice do lado direito de cada linha. Impressas todas as linhas, a função imprime os índices inferiores das colunas, na parte de baixo do tabuleiro.

3.5. realocaTab

Essa função foi feita para redimensionar o tabuleiro. Ela possui 4 chaves, duas para acrescentar linhas e duas para acrescentar colunas. As linhas e colunas são aumentadas uma a uma, exceto na primeira jogada, uma vez que depois dela o tabuleiro ganha duas linhas e duas colunas. Porém, isso é feito sem auxílio de funções, *hard-coded*.

Na primeira chave, é o caso em que uma pessoa coloca uma peça na primeira linha. Nesse caso, é preciso criar uma linha abaixo e “puxar” todas as peças para baixo, no tabuleiro, para que a linha superior não fique limitada e os jogadores possam fazer jogadas nela. Para tanto é utilizado um laço.

Na segunda chave, é o caso em que o jogador insere uma peça na última linha. Nesse caso, não há necessidade de “puxar” as peças para baixo: apenas inserir uma nova linha abaixo já é o suficiente para não deixar o tabuleiro limitado inferiormente. Portanto, é inserida uma nova linha e suas posições são inicializadas com espaço, para não invalidar as verificações do programa e o tabuleiro ser impresso da forma correta.

A terceira chave é usada quando o jogador insere uma peça na primeira coluna. Quando isso ocorre, é preciso inserir uma nova coluna e puxar todas as peças para a direita, para que a primeira coluna não fique inutilizada e os jogadores possam utilizá-la para efetuar suas jogadas. Novamente, é utilizado um laço para “puxar” todas as peças do tabuleiro para a direita.

Por fim, a última chave é usada caso o jogador insira uma peça na última coluna. Caso isso ocorra, não é preciso “puxar” as peças para a direita, e sim apenas inserir uma nova coluna. Isso é feito e as posições da nova coluna são inicializadas com espaços.

3.6. verificaJogada

No jogo Qwirkle são necessárias muitas verificações para ter certeza de que uma jogada é possível. A maior parte delas foi feita na função verificaJogada, enquanto algumas outras foram feitas no próprio corpo da função “jogada”.

Inicialmente, para que uma peça possa ser colocada em uma posição, as quatro posições ao redor dela - à direita, à esquerda, acima e abaixo - devem ser compatíveis com a peça em questão. Para fazer tal conferência, foram usadas duas variáveis: “cont” e “vazia”.

Uma peça é compatível com a posição ao lado quando sua letra ou seu número é igual ao da peça ao lado, ou então se a posição ao lado está vazia. Caso

alguma dessas condições seja verdadeira, o “cont” é incrementado. Se a posição estiver vazia, “vazia” também é incrementado. Isso é repetido com as 4 posições ao redor.

Existe ainda mais uma verificação necessária, para ver se a jogada pode ser realizada: caso uma peça igual a que o jogador está tentando inserir já esteja posicionada na mesma fileira de peças (seja ela a mesma linha ou coluna sequenciais), a jogada não pode ser efetuada. Novamente, a função confere as fileiras de peças adjacentes à posição alvo da jogada: coluna acima e abaixo e linha à esquerda e à direita. Se uma peça igual for encontrada, a função retorna 0 (falso) imediatamente.

Feitas todas as verificações, caso o “cont” seja igual a 4, quer dizer que a peça é compatível com todas as posições ao seu redor. No entanto, “vazia” não pode ser igual a 4 (exceto na primeira jogada, que não usa essa função), porque, caso seja, o jogador está tentando jogar a peça numa posição sem nenhuma peça ao redor, e isso não é permitido pelas regras do jogo. Portanto, se “cont” for igual a 4 e “vazia” for diferente, a função retorna 1 (verdadeiro). Caso contrário, retorna 0.

3.7. posJogada

Esta pequena função é utilizada ao final de cada jogada, recebendo as peças que foram utilizadas e transformadas em espaços no baralho do jogador e trocando-as por peças válidas que foram recebidas na rodada.

3.8. punktuation

A função punktuation realiza a conferência da pontuação que não é feita na função jogada. Essa última, confere a pontuação referente às linhas e colunas adjacentes à linha ou coluna da jogada principal, enquanto a punktuation confere a pontuação referente à linha ou coluna na qual o jogador está colocando suas peças.

Se o jogada é igual a 1, quer dizer que o jogador não jogou nenhuma peça (a variável jogada é inicializada como 1), e portanto sua pontuação é 0. Caso seja 2, quer dizer que o jogador fez uma jogada e o valor da variável jogada foi incrementado. Nesse caso, basta conferir as linhas e colunas adjacentes à posição da jogada - isto é, coluna de peças abaixo e acima, e linha à direita e à esquerda - realizada para computar os pontos.

Quando o jogador realiza duas jogadas ou mais, as conferências são mais complexas, e por isso foram divididas entre as funções “punktuation” e “jogada”. Como mencionado anteriormente, quando são feitas duas ou mais jogadas, a função punktuation confere apenas a pontuação referente à linha ou coluna principal da jogada.

3.9. jogada

Esta é a função em que, de fato, a jogada acontece. Inicialmente, o tabuleiro é impresso a cada rodada, juntamente do jogador em que está a vez e suas peças disponíveis, além das instruções para cada jogada. Em seguida, o jogador deve digitar o comando que deseja fazer. Uma variável (“movimento”) recebe o comando por partes: primeiro jogar (ou “j”), passar (ou “p”) ou trocar (ou “t”), segundo a peça a ser jogada, e por fim as posições em que se deseja ser feito movimento. O usuário pode tanto digitar a letra das peças maiúsculas ou minúsculas, uma vez que o programa converte as letras minúsculas para maiúscula com a finalidade de facilitar a verificação.

As primeiras verificações giram em torno de conferir se foram digitadas entradas na ordem correta e também se o jogador tem em suas “mãos” a peça escolhida. Em seguida, o programa corrige a variável de entrada da posição a ser jogada convertendo a entrada de char para int, a fim de se fazer as verificações necessárias. Algumas manipulações são feitas para índices de linha ou coluna de 2 algarismos, visto que cada “char” representa um algarismo do índice.

Em seguida, começam as verificações dos espaços escolhidos. Para tanto, são utilizadas as variáveis “jogadasTotais” e “jogada”, que recebem o número total de jogadas do jogo até aquele ponto e o número de jogadas feito pelo jogador daquela rodada até o momento, respectivamente. Caso seja a primeira jogada do jogo todo, a função de verificar jogada não é utilizada e o tabuleiro é excepcionalmente expandido em todas as direções.

Caso não seja, a função de verificar jogada será utilizada. Se a jogada for a primeira do jogador, e a jogada escolhida for válida, conforme a função, o tabuleiro é atualizado com a peça escolhida, e a linha e coluna escolhidas são guardadas. Se for a segunda, compara a linha atual à linha anterior e faz o mesmo com a coluna e armazena a informação de que o jogador está jogando na mesma linha ou coluna. Se não estiver jogando numa mesma linha, nem mesma coluna, a jogada é inválida e o jogador deve tentar novamente. Por fim, se a jogada for a terceira ou alguma seguinte, compara-se a linha ou a coluna, que devem ser iguais as das jogadas anteriores, o que é garantido pelas informações armazenadas.

Durante esse processo, uma flag é “levantada” caso a jogada tenha ocorrido de forma correta. Essa flag permite a locomoção das peças no tabuleiro e sua expansão. A flag foi especialmente usada para o tabuleiro não expandir caso a jogada seja inválida. Caso o número total de jogadas atinja o valor de 108 (número total de peças) a função é interrompida e o jogo termina.

Além de todas as funcionalidades da jogada (trocar, jogar ou passar, e algumas verificações de jogada), essa função possui algumas verificações de pontos. Uma parte da verificação dos pontos é feita de maneira simultânea, ao longo da função jogada, enquanto a parte suplementar é feita pela função

“punktuation”. Nesse sentido, o jogador poderá ver parte de seus pontos atualizando em tempo real, enquanto outra parte só será mostrada quando chegar sua vez novamente, porém as pontuações são computadas corretamente.

3.10. cheatMode

A função de trapaça permite que o jogador jogue uma partida de maneira semelhante a uma partida comum, porém tem acesso a qualquer peça que desejar utilizar. Dessa forma, optamos por remover a opção de troca de peças, e também, o programa deixa de imprimir a troca como uma das opções.

Para tanto, uma variável auxiliar para as peças é inicializada. Essa variável recebe a letra e o número da peça escolhida pelo jogador e o programa verifica se a escolha foi de uma letra entre a e f e de um número entre 1 e 6.

A função corre de maneira análoga à função de jogada. Entretanto, por não conter a opção de troca de peças, não contém linhas que tornavam as peças usadas da “mão” do jogador espaços vazios e também não apresenta a comparação da string a “trocar” ou “t”.

A função chega a um fim após 108 jogadas no total, como se simulasse o número máximo de peças do jogo. Entretanto, peças iguais podem ser utilizadas inúmeras vezes ao decorrer do jogo, como no jogo de exemplo composto pelo professor.

4. Função main

A função main constitui o corpo do jogo. No começo da função, as variáveis necessárias são inicializadas e o tabuleiro é alocado com 108 linhas e 108 colunas, visto que ele nunca extrapolaria essas dimensões. O grupo optou por fazer dessa forma, sem utilizar o realloc, porque o programa apresentou muitos problemas de segmentation fault quando eram utilizadas alocações dinâmicas com realocações recorrentes.

O jogo é mantido dentro de um laço que não é quebrado até que sejam efetuadas 108 jogadas, isto é, até que todas as peças sejam jogadas. O programa recebe do jogador o modo de jogo desejado, por meio de um getc, na variável “modo”. É perguntado ao usuário se ele deseja jogar no “cheat mode” e, caso ele insira a letra S (maiúscula ou minúscula), o jogo é iniciado no modo cheat. Se o usuário inserir a letra N (também maiúscula ou minúscula), o modo normal de jogo é iniciado.

Por fim, a função verifica se houve algum vencedor ou se o jogo termina empatado. Ademais, imprime na tela a pontuação de todos os jogadores da partida e desaloca a memória utilizada para jogadores e peças.

Link para o repositório no github:
https://github.com/drunknosferatu/qwirkle/tree/v1

Links para os vídeos

https://drive.google.com/file/d/15ZHZcdmOCCUeiSAXlZ5ZYV6yG2F2TU6Y/view?usp=sharing

https://drive.google.com/file/d/1KCaGsodDo5tTp3KShnxYQLBjyWNz_RQs/view?usp=sharing