UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

# INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO - TRABALHO 05: "QWIRKLE"

Lucas Alves Roris
11913771
Isabella Ferreira Lopes
10872281

SÃO CARLOS
Julho de 2020

# 1. INTRODUÇÃO À ATIVIDADE

Nesta tarefa, foi proposto o desenvolvimento do jogo “Qwirkle”, uma estrutura baseada em blocos cujo objetivo é a combinação de equivalentes letras ou números. Para o posicionamento de peças, esse jogo se vale de minuciosas regras que deveriam ser respeitadas no desenvolvimento do programa. Para esse trabalho, os principais recursos da linguagem C utilizados foram ponteiros, condições e structs. Os requisitos deste exercício foram:

- Tabuleiro dinâmico, que aumenta em uma linha e/ou coluna quando o usuário insere uma peça na extremidade;
- Bloqueio de entradas e posicionamentos inválidos dos jogadores;
- Apresentação das situações atuais do tabuleiro e das pontuações;
- Possibilidades de troca de peças e passe de turno;
- Possibilidade de modo “Cheat”.

## 2. LÓGICA

Os dois principais desafios quanto à lógica do programa foram a dinamização do tabuleiro e a definição de regras. Do primeiro problema, foi decidido que a matriz que representa o tabuleiro seria atualizada de cinco em cinco linhas/colunas, para evitar uma dinamização a cada jogada, porém, é apresentado ao usuário somente o meio da matriz, e ele vê o tabuleiro da maneira que foi requisitada: com apenas uma borda de elementos vazios. Os outros elementos ficam "escondidos" e entram na apresentação do tabuleiro no momento adequado. Em relação à definição de regras, foi feita uma função que identifica os seguintes parâmetros:

- Tentativa do usuário de inserir numa linha uma peça de letra ou número diferente do padrão daquela mesma linha;
- Tentativa do usuário de inserir numa coluna uma peça de letra ou número diferente do padrão daquela mesma coluna;
- Tentativa do usuário de jogar numa linha ou coluna diferente da que ele está completando desde o início de sua jogada;

Além disso, foram criadas unidades de tratamento de entradas inválidas fornecidas pelo usuário. Por exemplo, é bloqueada a tentativa de posicionar uma peça que não está entre as disponíveis para aquele jogador.

O programa foi fracionado em diferentes arquivos .C para o agrupamento de funcionalidades. Essas separações serão apresentadas a seguir.

## 3. DESENVOLVIMENTO

As funções do programa estão dispostas da seguinte forma dos arquivos.C:

- **ALOCACAO.C**

  Armazena uma função que libera a memória ao final do jogo, utilizando a função "free" para todos os ponteiros. Também contém as funções que verificam, a cada alocação ou realocação de ponteiros, se o retorno foi igual a NULL. Nesse caso, o programa é abordado e uma mensagem é apresentada ao usuário. São três funções, e cada uma analisa ponteiros de um dos tipos: ponteiro "dados", ponteiro "pecas" e ponteiro de ponteiro "peca".

- **JOGO.C**

Aqui ficam as funções que tratam da manipulação de peças e pontuação dos jogadores.

1. **AuxPontuacao**: a cada movimentação que o jogador faz, esta função verifica, com base na posição que foi ocupada, o total de elementos sequenciais acumulados e, portanto, a pontuação que o jogador adquiriu com aquela jogada;
2. **Pontuacao**: para cada linha e coluna da movimentação do jogador, esta função chama a AuxPontuacao para verificar os pontos, e incrementa a pontuação atual num ponteiro *Ptemp. Vale ressaltar que essas funções consideram as bonificações possíveis das regras do jogo (ex. se é completada uma linha ou coluna com seis elementos, o jogador ganha, além da pontuação comum, mais seis pontos de bônus);
3. **SortearPecas**: para cada vez que é necessário disponibilizar peças a um jogador, esta função utiliza o comando rand e a variável **PecasRestantes para sortear novas peças. Caso não haja mais peças para sorteio, é informado ao usuário;
4. **InicializarPecasRestantes**: para não sobrecarregar os comandos na função Main, foi criada uma função apenas para criar as peças do jogo, sendo que as letras das peças vão de A a F e os números de 1 a 6. Essas peças são armazenadas no ponteiro **PecasRestantes, e esse ponteiro será alterado ao longo do jogo;
5. **TrocarPeca**: quando o jogador escolhe trocar uma ou mais de suas peças, esta função é chamada, e o primeiro procedimento é fazer a leitura das peças que o jogador escolheu, que são armazenadas numa matriz. Em seguida, as peças inseridas são localizadas entre as peças do usuário através de dois laços For, e se

não forem localizadas, o usuário é informado de que digitou algo errado. Por fim, essas peças são substituídas por novas, através da função SortearPecas;

6. **Jogada**: após ler a peça que o jogador deseja inserir, esta função verifica se a peça faz parte da coleção desse jogador, e, se sim, armazena numa variável PecaJogada. Se o modo Cheat estiver ativado, não há essa verificação. Há os devidos tratamentos de erro de entrada. As coordenadas inseridas pelo usuário são traduzidas para as correspondentes na matriz do tabuleiro, e então é verificado se essas posições estão próximas da borda a ponto de ser necessária a dinamização do tabuleiro, e se sim, a função AtualizarTabuleiro é chamada. Em seguida, é conferido se a função MovimentoValido aponta 1, pois isso significa que não há problemas com as regras, e os devidos posicionamentos no tabuleiro são feitos, inserindo a peça na coordenada. Se o modo Cheat não estiver ativado, é diminuído o número de peças restantes do jogador;
7. **Instrucoes**: printa algumas instruções para bom uso do jogo ao usuário antes do início da partida.

• **MOVIVALIDO.C**

Nesse arquivo foram desenvolvidas as duas funções que verificam a validade dos movimentos solicitados pelos jogadores. Elas são descritas a seguir.

1. **AuxMovimentoValido**: esta função se vale das regras do jogo para comparar a posição escolhida com os seus arredores. São três blocos de verificação: (1) se na posição escolhida há uma sequência de números e se a peça escolhida se encaixa, (2) se na posição escolhida há uma sequência de letras e se a peça escolhida se encaixa e (3) se a peça adjacente está sozinha e se há alguma semelhança entre ela e a escolhida. É retornado 1 caso haja encaixe, e 0 no caso contrário;
2. **MovimentoValido**: primeiramente são verificadas duas condições básicas: se o valor da posição requerida pelo usuário é diferente de ‘ ‘ (espaço), pois isso automaticamente indica que o movimento é inválido, e portanto o retorno é 0, e se a jogada é 0 0 na primeira condição do tabuleiro, pois logicamente aquela logada é válida, e o retorno é 1. A partir daí, quatro condições If são utilizadas para verificar os elementos ao redor da posição inserida: acima, abaixo e dos dois lados. Para cada uma dessas verificações, é chamada a função AuxMovimentoValido, e uma variável flag armazena o retorno da função. Se ao

final das quatro verificações a flag for 1, o movimento é válido. Do contrário, não é.

• **TABULEIRO.C**

Armazena as funções relativas ao tabuleiro do jogo e à sua dinamização.

1. **DefinirCor**: Nessa função, um laço switch recebe o número correspondente a cada peça e, de acordo com ele, define a cor da peça com os respectivos códigos;
2. **AtualizarTabuleiro**: quando esta função é requisitada, ela primeiramente faz as devias alocações e verificações do ponteiro **Tabuleiro, e em seguida utiliza as fórmulas abaixo para transferir todos os elementos para as respectivas posições:
   - **Nova Linha = Linha Anterior + (Tamanho Novo – Tamanho Anterior)/2**
   - **Nova Coluna = Coluna Anterior + (Tamanho Novo – Tamanho Anterior)/2**

   Por fim, os novos elementos vazios são preenchidos com ‘ ‘ (espaço);
3. **PrintarTabuleiro**: utilizando padrões para inserir as barras e espaçamentos dos elementos, esta função printa o tabuleiro, quebrando com ‘\n’ a cada fim de linha, e insere os respectivos elementos entre as separações.

O arquivo struct.h armazena as structs “peca” (elementos letra e número) e “dados” (elementos nome, quantidade de peças, peças e pontos).

Na função principal, é inserida a função srand para a manipulação de escolhas aleatórias, e em seguida são lidos os dados do usuário: número de jogadores, nome de cada um deles, opção de Cheat Mode. As variáveis para cada um desses dados são inicializadas e, no caso dos ponteiros, são chamadas as funções de verificação. São mostradas ao usuário algumas instruções de jogo, e o tabuleiro e as peças são inicializados. Há um laço que mantém o jogo, que é mantido enquanto o ponteiro *qPecasRestantes é diferente de 0. Dentro delem há um laço For que mantém a sequência dos jogadores, e ainda dentro desse, há um outro laço While que mantém a jogada, e é quebrado quando o jogador passa a vez. São apresentadas ao jogador as opções, as peças disponíveis e o tabuleiro. Para cada opção escolhida são feitos os tratamentos de erro e são chamadas as correspondentes funções. Quando o jogo acaba, é chamada a função que libera os ponteiros.

## 4. FINALIZAÇÕES

Para aprimorar a visualização do jogo, a cada função executada foi dado um comando para limpar a visualização da tela. Esse comando é o System(“clear||cls”), que funciona no sistema operacional Linux e no Windows.

Foi adicionado um comando que anuncia o vencedor ao fim do jogo, que é aquele que obteve maior pontuação.

As aplicações das regras do jogo foram revisadas para a garantia de funcionamento das funções e da lógica.

## 5. INSTRUÇÕES PARA JOGAR

1. Insira os nomes dos jogadores que participarão;
2. Selecione sim (S) ou não (N) para a opção Cheat Mode, quando não há restrições de peças para inserir;
3. No seu turno, selecione passar para pular a vez, trocar para trocar uma ou mais (até seis) de suas peças, sendo que, nesse caso, sua vez será pulada, ou jogar para inserir uma peça;
4. Para jogar, na mesma linha, escreva a peça que quer colocar e a posição em coordenadas (linha coluna). As coordenadas estarão explícitas no tabuleiro. Tente posicionar o maior número de peças seguindo as regras:
    a. Numa linha com mais de um elemento, deve ser seguido o padrão de letra ou número daquela mesma linha;
    b. O mesmo se aplica para uma coluna;
    c. No caso de haver apenas um elemento, posicione uma peça que seja compatível com ele em linha ou coluna;
    d. Uma vez iniciada a colocação em uma linha ou coluna, deve-se continuar na mesma até o fim do seu turno.
5. Cada elemento da linha ou coluna que você ajudar a completar vale um ponto. Ao completar uma linha ou coluna com seis elementos, são dados seis pontos de bônus;
6. O vencedor é aquele que, com o fim das peças, tiver mais pontos acumulados.

## 6. VÍDEOS

O grupo gravou uma vídeo-chamada de explicação de algumas resoluções do programa, e um segundo vídeo de apresentação do jogo. Abaixo, os links para esses vídeos.

SOLUÇÃO

APRESENTAÇÃO