Relatório de ICC1

Trabalho 05: Qwirkle

-Integrantes do grupo:

David Felipe Santos e Souza Dias. nUsp - 11800611

-Link para videos:

video de discussão:
https://drive.google.com/file/d/1eVPccLBlH6XY7D8jb-NcqZQ0QJlleru9/view?usp=sharing

video de execução:
https://drive.google.com/file/d/1BObzxp1l5bpdlaV5C0c-HWh30zCKqWW5/view?usp=sharing

-Inicialização:

Para que o jogo inicie basta colocar na linha de comando as cláusulas “make all” e “make run”, nessa ordem. Com apenas isso o jogo seria iniciado esperando o input com o número de jogadores.

-Jogabilidade:

Ao iniciar o programa, será pedido o número de usuários seguido dos nomes desses usuários. Então será perguntado se deseja usar o cheat mode, que pode ser respondido com ‘s’ e ‘n’. Após isso o jogo começa de fato, onde será pedido um comando o qual podem ser os comandos:

1. Jogar
   Deve ser escrito da seguinte forma: “jogar p1 x y”, onde p1 é a peça escolhida, x é a linha e y é a coluna que deseja colocar a peça.
2. Trocar
   Deve ser escrito da seguinte forma: “trocar p1”, onde p1 é a peça que deseja trocar por uma aleatória.
3. Passar
   Somente “Passar” acaba o seu turno.

-Funcionalidades:

- Checar movimentos inválidos segundo as regras do jogo.
- Fazer o tabuleiro ter tamanho variável segundo as localizações das peças jogadas.
  -Essa funcionalidade envolve deslocar todo o tabuleiro para algum lado caso a linha ou coluna tenha como valor do input do jogador negativo.
- Checar por pontos e quem ganhou.

-Descrição da solução:

- **Structs:**
  A existência de structs foi vital para o programa, visto que boa parte de suas funcionalidade giram em torno de duas: “peca” e “jogador”.
  A primeira se mostrou preferível a simplesmente criar uma string de dois caracteres pois o formato de struct facilita o entendimento do código, apesar de gerar algumas linhas a mais devido a impossibilidade de uso do comparador “==” em structs, sendo necessário igualar seus dados.

A segunda struct é intrínseca ao código, visto que armazena informações muito importantes e conectadas, como nome do jogador, peças que o jogador tem, pontos do jogador, quantidade de vezes que o jogador jogou nesse turno e jogadas anteriores desse jogador. O último se mostra de extrema importância no cálculo de pontos e invalidação de movimentos.

- **Variáveis Globais:**
  Algumas variáveis, por serem usadas por diversas funções do código, tiveram melhor funcionalidade sendo declaradas de forma global, como:
  1. npecas
  2. allPecas
  3. formatos
  4. cores
  5. cheatMode
  6. e o quarteto xMax, xMin, yMax, yMin

  A primeira é somente uma forma de saber o quão próximo o jogo está de acabar. Acaba sendo acessada por diversas funções por no primeiro turno existirem muitas regras diferentes .
  A 2, 3 e 4, apesar de não serem acessadas por muitas funções, fizeram mais sentido estando como globais, visto que não faria sentido passá-las como parâmetro para certas funções.
  A 5 é importante em diversas funções para saber quando poderá ser jogada qualquer peça, mesmo que o jogador não a tenha.
  E o quarteto final apenas diz quais números serão imprimidos ao se imprimir o tabuleiro. São acessadas por diversas funções e tem seus valores mudados pela função gameMove.

- **Funções:**
  Esse código utiliza diversas funções para imprimir informações a tela, checar possíveis erros na jogada e checar pontos. Entre essas são:
  1. Colorize
  2. GameState
  3. validMove
  4. validCommand
  5. gameMove
  6. randomizarPecas
  7. checkPoints
  8. fgetss*
  9. atoii*

  -Comentarei sobre apenas algumas mais importantes. As duas com o * são funções que tem o mesmo propósito que funções que já existem na biblioteca padrão de c, mas com funcionalidades específicas para esse código. A fgetss, analoga a fgets, recebe uma string e limpa o buffer e a atoii , analoga a atoi transforma os caracteres da linha e coluna que o jogador digitou ao jogar uma peça em números.
  -A função GameState imprime o tabuleiro para a tela, levando em consideração os valores máximos e mínimos descritos anteriormente.

-validCommand e validMove verificam, respectivamente, se o comando enviado pelo jogador está correto e se a jogada que o jogador deseja fazer é válida.
-A função gameMove tem duas funções principais: Controlar qualquer jogada que o jogador fizer e alterar os valores mínimos e máximos que serão imprimidos na tela. Ela usa as funções de validação para entrar em recursão e esperar um comando válido ou colocar as peças que o jogador quer nas posições que ele quer no tabuleiro.
-A randomizarPecas acaba por “colocar” todas as 108 peças no vetor allPecas e gerar o vetor formatos e o vetor cores.