Universidade de São Paulo

SSC0600 - INTRODUÇÃO À CIÊNCIA DE COMPUTAÇÃO
SSC0601 - LABORATÓRIO DE INTRODUÇÃO À CIÊNCIA DE COMPUTAÇÃO
TRABALHO 05

RELATÓRIO QWIRKLE

Prof.: Adenilso da Silva Simão
Fernando Cardoso Braghiroli Nº USP: 11800500
João Alexandro Ferraz Nº USP: 11800441

Descrição da solução:

Inicialmente, utilizamos o whatsapp para iniciar as discussões sobre o código e como iríamos prosseguir. Em seguida, nos reunimos algumas chamadas via google meets para tentar estabelecer uma resolução ao problema.

Descrição das funções:

LimpaBuffer

A função em questão desempenha o papel de evitar problemas em relação à leitura de dados, isto é, através de um loop while, com o caractere “c” != ‘\n’ e “c != EOF” (fim do que está sendo lido), há a garantia de que haverá a leitura até um caractere de “newline” (‘\n’) ser encontrado ou até acabar o que está sendo lido.

Acabou

Essa função tem o papel de contar quantas peças do array “qwirkle” foram mudadas para “00”, visto que, quando um usuário recebe a peça do array, a posição do array que foi inserida na mão do usuário é mudada para a string “00”. Através de uma variável inteira “cont”, a função verifica se há 108 strings “00”. Se houver, indica que não há mais jogadas a serem feitas, e o jogo acaba.

colocar_na_mao

Essa função, por sua vez, realiza o trabalho de colocar as peças do array na mão dos usuários que estão jogando o qwirkle. Através de um loop while, com um contador “i” que vai de 0 à 5, um número “k” é sorteado aleatoriamente, no “range” de 0 à 107 e, então, a peça correspondente a esse número no array é colocada na mão do usuário. Isso só acontece se a peça da posição “k” não tiver sido já substituída previamente, isto é, não valer “00”. Caso tenha o valor de “00”, um outro número “k” é sorteado e a verificação ocorre novamente. Por fim, se a peça da posição “k” não valer “00”, por meio da função “strcpy” a peça do array é copiada para a mão do jogador e , em seguida, esta posição do array recebe a string “00”, indicando que houve uma mudança. Há o incremento do contador ao realizar essa etapa e , quando este valer 6, o loop while não ocorrerá novamente, indicando que todas as peças foram colocadas com sucesso em todos os espaços da mão dos jogadores.

expandir

Essa função é primordial para o funcionamento do tabuleiro e foi o maior motivo de dificuldade do grupo para fazer funcionar em conjunto com as outras funções. Ela (a função) recebe como parâmetros 2 ponteiros, e 2 ponteiros de ponteiros. Primeiramente, a função verifica se o índice (index) da linha está na última posição do tabuleiro, visto que, se ocorrer isso, a expansão é necessária para que o jogador possa visualizar mais campos do tabuleiro e assim realizar mais jogadas. Dentro dessa mesma verificação (if), também vê-se se o index digitado foi na primeira linha, pois, se for e não houver a expansão, o jogador não conseguirá jogar em linhas acima desta, apenas para os lados. Após a condição ser cumprida `if(index == **y_y - 1 || index == 0)` , tem-se que uma variável de controle para a linha muda de valor, como se fosse uma flag (*tmpy). Essa variável serve para marcar se houve a expansão de linha.

Então, após a mudança da flag, tem-se o incremento da variável que controla o tamanho das colunas (**y_y). Após isso, verifica-se se o index == 0 e , se essa condição for cumprida, as peças do tabuleiro irão “se deslocar” para uma linha abaixo, visto que se isso não ocorrer, o tabuleiro apenas se expande para baixo, e as peças continuam no index = 0, não podendo ser colocado novas peças em linhas acima.

Em seguida, há a verificação para as colunas, seguindo o mesmo padrão da lógica anterior: se o index2 (índice da coluna) for na posição final, ou inicial do tabuleiro, a variável de controle de coluna é incrementada (*tmpx) indicando que houve a expansão da mesma e , então, a variável **x_x é incrementada, fazendo a coluna consequentemente expandir, visto que “**x_x” é o tamanho da coluna possível para o jogador.

Ainda há o mesmo procedimento feito para as linhas, isto é, a comparação para ver se a coluna digitada foi a primeira possível do tabuleiro. Se essa condição for cumprida, ocorre o rearranjo das peças no tabuleiro, com a cópia de espaços vazios nas posições que foram deslocadas, visto que assim não se tem o perigo das peças aparecerem duplicadas no tabuleiro.

Por fim, como última comparação, verifica-se se a última linha e coluna do tabuleiro estão preenchidas por alguma peça e, se estiverem, ocorre tanto o incremento da variável “**y_y” quanto da “**x_x”.

Portanto, essa função realiza a expansão do tabuleiro de qwirkle.

menu

Essa função desempenha o papel de ser o menu seletor do programa, isto é, se o usuário que está jogando deseja trocar uma peça, jogar, ou passar sua vez.

A variável do tipo char v[12] é lida através de um fgets e, em seguida, é feita a remoção do caractere ‘`\n`‘ de sua string, visto que é necessário para fazer a comparação do que é colocado pelo usuário com os comandos do programa.

Se o usuário digitar “passar”, a função retorna o inteiro “1”. Se o usuário digitar “jogar”, a função retorna o inteiro “2”, e se o usuário digitar “trocar”, a função retorna o inteiro “3”. Esses números servem para serem utilizados em outras funções de acordo com o que o usuário deseja.

Por fim, se o usuário não digitar nenhum desses comandos, a função entra em um loop while, imprime uma mensagem de erro, e faz-se necessário digitar o comando certo para fazer desempenhar seu papel. Só há a saída desse loop quando o usuário digita o comando certo.

trocar

Essa função tem o papel de fazer a troca de peças da mão do usuário, caso for digitado “trocar” no menu.

Primeiramente, um valor inteiro é atribuído de maneira aleatória para a variável “k”, visto que essa irá indicar a posição no array que contém as peças do jogo para realizar a troca. Posteriormente, ao valor ser atribuído, através de um “strcmp”, há a verificação na posição “k” do array que contém as peças do jogo está atribuída a string “00”. Caso essa string esteja atribuída, um novo “k” será escolhido de maneira aleatória para fazer novamente a comparação. Enfim, se for != “00”, uma variável chamada subst irá receber a peça da posição “k” do array.

Em seguida, o usuário irá digitar a peça que deseja trocar e , então, o programa entra em um loop while(1). Se a peça digitada existir na mão do usuário, um for loop é acionado para fazer uma varredura no array “qwirkle” e a primeira vez que ele encontrar uma string “00”, a peça da mão do jogador vai para a posição “v” do qwirkle, o jogador recebe a peça “subst” (foi pega no início da função) e uma flag é mudada para o valor “1”, indicando que houve a troca com sucesso.

Se o programa verificar que a flag vale “1”, então é impresso uma mensagem indicando que houve a troca e a função acaba.

Caso o contador “i” atinja o valor “6”, isso indica que a contagem excedeu o tamanho da capacidade das peças da mão do jogador e , portanto, não há determinada peça na mão deste, indicando a troca não será possível. Caso essa condição se cumpra (if == 6), uma mensagem de que não foi possível trocar será impressa e a função será terminada.

movimento

Essa função trabalha em conjunto com a **trocar**, visto que é ela quem decide os valores de “tempx” e “tempy”.

A função começa verificando se o valor do índice da coluna é igual ao de *tempx (passado como ponteiro no argumento da função). Se essa condição for cumprida, *tempx valerá “-1” e a função será encerrada, retornando 0.

Essa mesma verificação é feita para a coluna e, caso nenhuma dessas duas verificações sejam cumpridas, a função retorna “1”.

Caso o número de jogadas seja maior do que 2, e se o índice da linha for == *tempx, o programa retornará “0”. A mesma coisa para a coluna. Se nenhuma dessas 2 condições forem cumpridas, o programa retorna “1”.

Por fim, para esclarecer, essa função tem como objetivo, por meio das variáveis *tempy e *tempx, fazer com que o usuário só seja capaz de jogar na mesma linha ou coluna após a segunda jogada realizada em seu turno.

jogada

Essa função é a principal do programa: por meio dela, as jogadas são feitas no tabuleiro, a função menu é utilizada, e a grande maioria das outras também. É como o coração do programa.

Em seu início, ocorre o print da mão do jogador que está jogando, o print do tabuleiro e a chamada da função **menu**. Como o usuário só pode trocar de peça no início de sua jogada, o grupo optou por colocar essa opção no começo de seu turno, não sendo possível realizá-la posteriormente.

Se o usuário optar por trocar as peças, há a entrada em um loop e só haverá a saída desses quando for escolhido não trocar mais peças.

Posteriormente, se o usuário escolher jogar, haverá a entrada em um loop “while(a==2)” e 2 flags são declaradas “achou” e “correta”. É perguntado qual a peça e a posição que o usuário deseja jogar e , então, é feito uma comparação de se a peça que o usuário digitou existe na mão do usuário.

Caso essa peça exista na mão, a flag “achou” mudará para “1”, indicando que ela existe. Se “achou” for == 0 ou não houver um espaço vazio para jogar, uma mensagem de erro será impressa. Caso contrário, o tabuleiro recebe a peça, ocorre a verificação para ver se a jogada e o movimento são válidos e então, é copiado a string “00” para a mão do jogador, indicando que essa peça foi removida, e a flag “correta” é mudada. Se essa for a primeira jogada, tempx recebe index e tempy recebe index2.

Agora, se a jogada não for permitida ou o movimento == 1, haverá a impressão de uma mensagem de erro na tela e , em seguida, onde a peça havia sido inserida será preenchido com espaço. O programa perguntará se o usuário quer jogar novamente e, se deseja, sua mão é impressa, e o retorno ao loop é feito sem problemas. Caso o usuário não deseja, há um loop while que faz com que as peças que valem “00” na mão do usuário sejam substituídas por novas, vindas diretamente do array qwirkle.

jogada_permitida

Essa função verifica se o usuário pode inserir uma peça em determinado local, respeitando as regras da lógica do qwirkle.

Uma série de comparações é feita para verificar se a jogada é possível naquela linha (index) ou coluna (index2) passadas como parâmetro da função.

Basicamente, há 2 maneiras de raciocínio:

1 - Verifica-se se há peças repetidas na mesma linha ou coluna: isso é feito através da comparação →

```
while(strcmp(jogo[row][col+k],"  ")!=0 && col+k<=x_x+1){
    if(strcmp(jogo[row][col],jogo[row][col+k]) == 0){
      flag = 0;
      return flag;
    }
k++
```

} (O mesmo raciocínio é feito para as linhas)

2- Verifica-se se há um espaço vazio em um lado, e se há peça no outro, como no exemplo →

```
while(col+i <= x_x+1 && strcmp(jogo[row][col+i],"  ")!=0){
    if(jogo[row][col][0] == jogo[row][col+i][0] || jogo[row][col][1] == jogo[row][col+i][1] ){
      flag = 1;
      if(strcmp(jogo[row][col+i+1],"  ")==0){
        return flag;
      }
    }
```

3 - Se a jogada for possível, retorna-se a flag valendo 1. Caso não for, a flag é retornada valendo 0.

computar_pontos

Essa função desempenha o papel de contar os pontos dos jogadores após as jogadas serem realizadas. Especificamente, essa função o grupo encontrou extrema dificuldade em realizá-la de maneira efetiva e, após inúmeras tentativas, decidimos deixar do jeito que está sendo entregue no programa.

As comparações dessa função funcionam de modo semelhante à jogada_permitida, sendo que, ao encontrar uma peça, um contador começa a ser incrementado e os pontos começam a ser contados. Se a peça não for mais encontrada, há a saída do loop while da função e, por conseguinte, outro loop começa.

Dentro de cada um desses loops, há a condição de (if i == 5). Isso significa que foram contadas 5 peças (excluindo a que foi colocada por último) e , então, o usuário fez um “qwirkle” e sua pontuação é incrementada em 6 pontos.

Por fim, após realizar todas as verificações, a função retorna a quantidade de pontos feitas pelo usuário.

trapaca

Essa função é utilizada quando o modo trapaça está ativado. Ela funciona de forma semelhante à função **jogada,** porém possibilita o jogador escolher qualquer peça do jogo para jogar, diferentemente da **jogada** que apena possibilita peças que estão na mão do usuário. Para isso não há mensagem de erro quando o jogador escolhe uma peça diferente das suas, mas as regras checadas por **jogada_permitida** e **movimento** ainda devem ser respeitadas.

A lógica é semelhante quase a função inteira, exceto quando uma peça não é achada na mão do usuário. Se a peça não for achada, não é problema, visto que basta apenas serem respeitadas as regras do qwirkle.

print

Essa função é responsável por simplificar a impressão do tabuleiro para o usuário, utilizando as dimensões estabelecidas pela função expandir (x_x e y_y).

Como jogar

1 - Para jogar, o usuário deve digitar “jogar” , pressionar a tecla “enter” e , em seguida, digitar a peça e a posição que deseja;

2 - Para trocar, o usuário deve digitar “trocar”, pressionar a tecla “enter” e, em seguida, digitar a peça que deseja ser trocada;

3 - Para passar, basta o usuário digitar “passar” e teclar “enter”;

4 - No modo cheat, cabe ao usuário colocar apenas peças que pertencem ao “qwirkle”, visto que você está como administrador do jogo;

5 - Se o usuário digitar algum comando errado no menu, cabe a ele, posteriormente, digitar o certo. Caso contrário, ficará preso em um loop para sempre.

6 - Se atente às linhas de instruções que aparecem durante o jogo. Seja após uma jogada errada, ou para realizar uma jogada. Elas devem ser cumpridas com rigidez.

7 - Bom jogo!

Vídeo comentando o código:
https://drive.google.com/file/d/1WbmvZyxiB8-gkNCBFz6kprwiooeClzgl/view?usp=sharing

Vídeo testando o código:
https://drive.google.com/file/d/1VbHBYtkNB-6_Nbc1mVrvQG3vhE6s6nza/view?usp=sharing