PONG HAU K'I

Trabalho de Computação I

Integrantes:

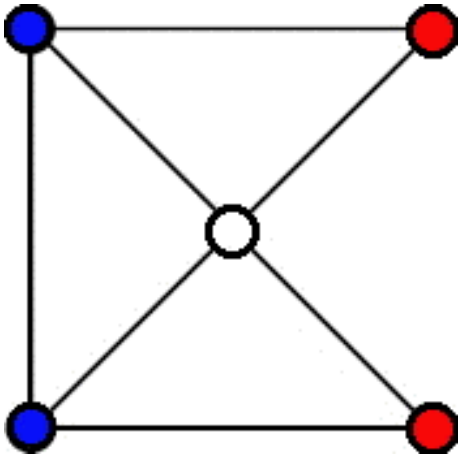
- Pedro Henrique Costa de Gois (DRE:121096287)
- Manoel Marcelo da Silva (DRE:121088349)
- Lucas de Lyra Monteiro (DRE:121039714)

SUMÁRIO

O QUE É PONG HAU K'I

"Pong Hau K'I" é um jogo tradicional oriental para dois jogadores, originado em Cantão, na China, e possui diversos outros nomes, como "Ou-Moul-ko-no" na Coréia e "Ferradura" em alguns países europeus. Ele é considerado um jogo infantil por sua facilidade de jogar.

Esse jogo de tabuleiro é bem simples, podendo ser reproduzido apenas com um papel. Os jogadores possuem duas peças, e só podem movimentar apenas uma peça por vez, seguindo as regras. O tabuleiro do jogo tem 5 vértices e 7 arestas que representam, respectivamente, as casas onde as peças podem ficar e o caminho entre essas casas. O objetivo do jogo é deixar seu adversário sem movimentos, travando-o em algum canto do tabuleiro.



Tabuleiro padrão de Pong Hau K'I

REGRAS DO JOGO

As regras de Pong Hau K'i são bem simples:

- O tabuleiro tem que começar com o centro vazio e com as peças iniciais posicionadas como acima.
- O jogador que começa é decidido aleatoriamente.
- Cada jogador só possui duas peças.
- Só é possível mover sua própria peça.

- A cada turno só é possível mover uma peça.
- Não é possível pular peças nem trocar as peças, ou seja, só é possível mover para espaços vazios adjacentes.
- Ganha quem deixar o adversário sem movimentos válidos.

OBJETIVOS DO TRABALHO

O nosso objetivo é implementar o jogo Pong Hau K'i , fiel ao jogo real, em linguagem C de forma acessível para todos os usuários, utilizando somente a biblioteca padrão da linguagem de programação C.

NOSSAS PREMISSAS

O programa deve explicar ao usuário não só o próprio jogo, como ao disponibilizar e mostrar a regra dele, mas também explicar o que está acontecendo de errado, por exemplo ao indicar que uma peça é inválida ou que um movimento é inválido.

Além disso, o programa deve ser autoexplicativo e seguir as boas práticas de programação, como nomes elucidativos para funções e variáveis. Além disso, o jogo deve ser fácil e acessível.

MANUAL DO USUÁRIO

Antes de tudo, como desejamos uma implementação fiel do jogo, o usuário deve respeitar todas as regras originais do "Pong Hau K'I", sendo o jogador 1 com as peças P e p e o jogador 2 com as peças B e b.

1.Compilando o Jogo

Como o nosso jogo foi feito usando a linguagem C, a portabilidade é garantida, logo é só compilar o arquivo .c do jogo na máquina escolhida para criar um executável funcional.

Para compilar execute o terminal na pasta em que está o .c e execute:

“gcc -o nome_do_arquivo nome_do_arquivo.c -Wall -ansi -pedantic”

Na pasta haverá um executável com o nome do arquivo e é só rodar para jogar, ou execute no terminal após compilar:

./nome_do_arquivo

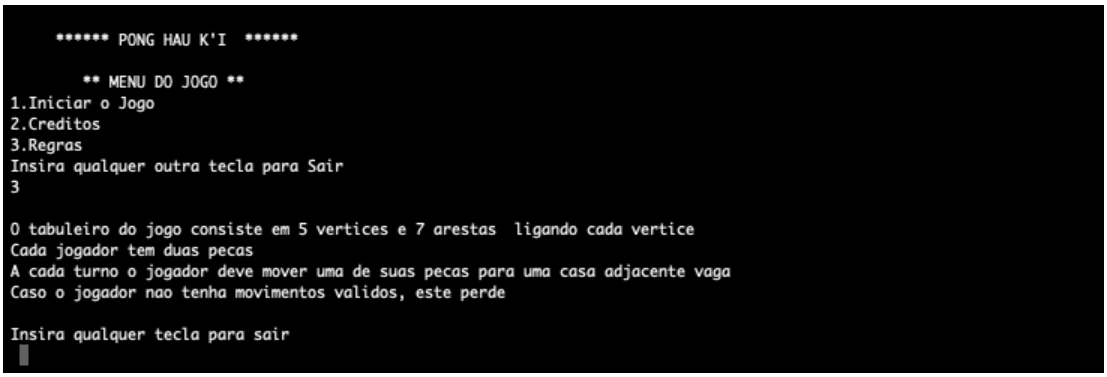
2.Menu

Ao iniciar o executável, haverá a seguinte tela:

Este é o menu, insira as teclas desejadas para fazer o que necessita.

Por exemplo, ao selecionar 2, você verá os créditos, e para retornar ao menu é só inserir qualquer tecla, como visto abaixo:

Ou caso deseje ver as regras do jogo, selecione 3, e para sair ao menu insira qualquer tecla:



Para sair do jogo completamente, digite qualquer tecla que não foi informada ao menu para sair:

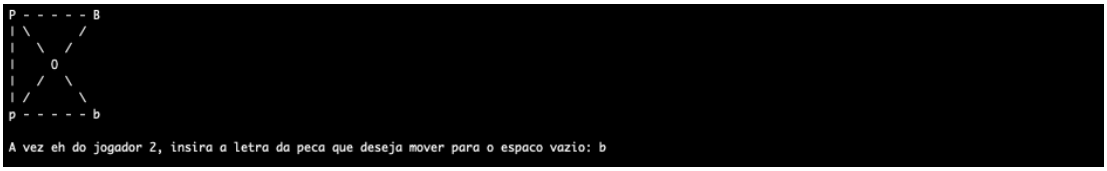
E por fim para começar o jogo, digite 1.

2.0 Jogo

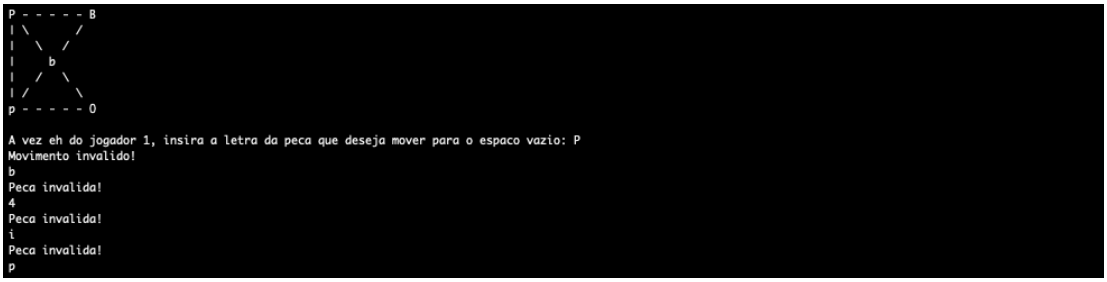
Ao começar o jogo, selecionando 1 no menu, é escolhido aleatoriamente o jogador que começa.

ou

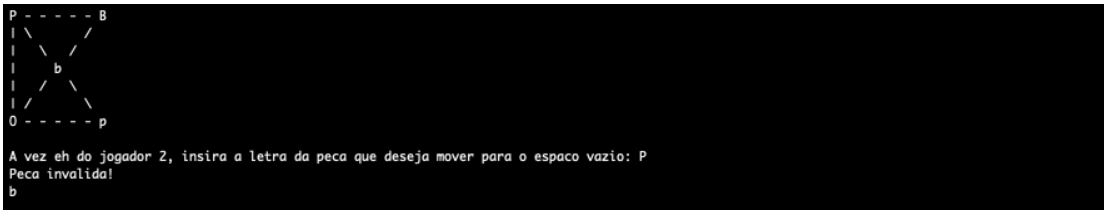
Após ver quem começa, insira qualquer tecla com enter para começar. E assim, você verá o tabuleiro inicial e um comando para mover a peça do jogador que começa:

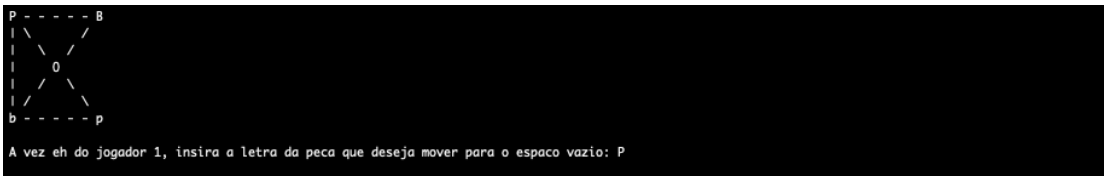


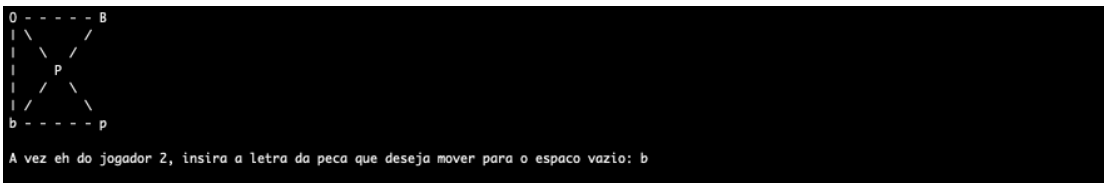
Caso seja inserido alguma peça inválida ou movimento inválido, será dado o erro respectivo, e coloque uma peça válida para movê-la e continuar o jogo:

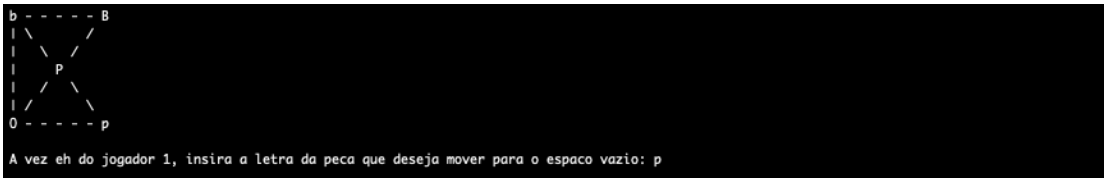


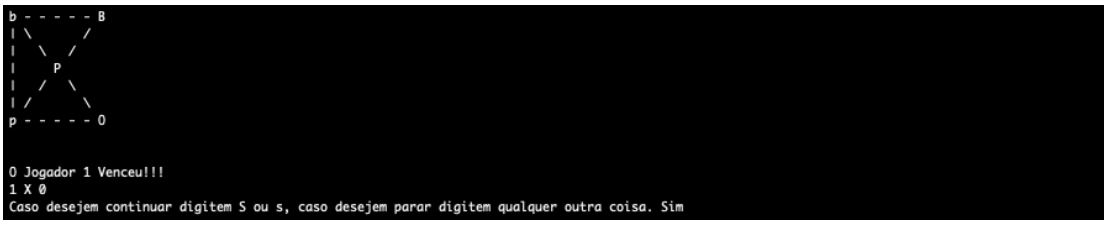
O jogo continuará até que alguém vença:











Quando alguém vence, é mostrado o placar desde quando começou o jogo e será perguntado se deseja jogar novamente, caso digite qualquer coisa que inicie com S ou s, irá começar outro jogo. Caso digite qualquer outra coisa, voltará ao menu e o placar será resetado.

Este é um exemplo de segunda partida:

```
O jogador 1 comeca o jogo com as pecas P e p!
O jogador 2 tem as pecas B e b

Aperte qualquer coisa para comecar!
```

```
P - - - - - B
| \       /
|   \   /
|     O
|   /   \
| /       \
p - - - - - b

A vez eh do jogador 1, insira a letra da peca que deseja mover para o espaco vazio: p
```

```
P - - - - - B
| \       /
|   \   /
|     p
|   /   \
| /       \
O - - - - - b

A vez eh do jogador 2, insira a letra da peca que deseja mover para o espaco vazio: b
```

```
P - - - - - B
| \       /
|   \   /
|     p
|   /   \
| /       \
b - - - - - O

A vez eh do jogador 1, insira a letra da peca que deseja mover para o espaco vazio: p
```

```
P - - - - - B
| \       /
|   \   /
|     O
|   /   \
| /       \
b - - - - - p

A vez eh do jogador 2, insira a letra da peca que deseja mover para o espaco vazio: P
Peca invalida!
b
```

```
P - - - - - B
| \       /
|   \   /
|     b
|   /   \
| /       \
O - - - - - p

A vez eh do jogador 1, insira a letra da peca que deseja mover para o espaco vazio: P
```

```
O - - - - - B
| \       /
|   \   /
|     b
|   /   \
| /       \
P - - - - - p

A vez eh do jogador 2, insira a letra da peca que deseja mover para o espaco vazio: B
```

```
B - - - - - O
| \       /
|   \   /
|     b
|   /   \
| /       \
P - - - - - p


O Jogador 2 Venceu!!!
1 X 1
Caso desejem continuar digitem S ou s, caso desejem parar digitem qualquer outra coisa. nao
```

Ao retornar ao menu, é possível sair do jogo:

E este é o jogo completo de Pong Hau K'I!

COMENTÁRIOS DOS DESENVOLVEDORES

Para melhor explicar o desenvolvimento do jogo, iremos mostrar todas as funções que estão no jogo e nossos aprendizados com elas.

- `printa_matriz(char matriz[LINHA][COLUNA])`

Nela o conhecimento de for-loop foi essencial para imprimir a matriz(tabuleiro) na tela, a função itera por todos os caracteres de uma linha e todas as linhas de uma matriz (recebida) e imprime cada caractere, sendo extremamente necessário para imprimir o tabuleiro

- `PONTO procura(char matriz[LINHA][COLUNA], char peca)`

O conhecimento de estruturas foi fundamental para retornar dois valores, a posição vetorial x e o y do que estamos procurando em uma matriz dada.

Inicialmente pensamos em retornar um vetor, porém uma implementação usando estrutura é muito mais rápida e limpa.

Outro fator importante foi o conhecimento de for-loop e condicionais if e operadores lógicos (==) para tentar encontrar um caractere numa lista de caracteres, fazendo uma varredura na função e retornar o index desse caractere. O jogo só funciona com essa função, sendo ela uma das funções mais importantes do jogo.

- `int peca_valida (char peca, int jogador)`

O conhecimento do uso de condicionais if foram essenciais para determinar se determinado input é válido, fazendo verificações com operadores lógicos == e || para determinar se uma peça existe e é de um determinado jogador. Muito útil para a validação no futuro.

- `void movimenta (char matriz[LINHA][COLUNA], char peca)`

Esta função necessita primordialmente da função `PONTO procura(char matriz[LINHA][COLUNA], char peca)` para encontrarmos a posição da peça que desejamos mudar para o espaço vazio, além da posição desta.

Logo, o conhecimento de modularização e funções para chamar uma função dentro de outra função foi extremamente necessário e importante para o movimento do tabuleiro, além disso, o conhecimento de matrizes para trocar as coordenadas do espaço vazio e da peca foi importante também.

- `void verfica_movimentos (char matriz[LINHA][COLUNA], char peca)`

Como na função `int peca_valida (char peca, int jogador)`, usamos o conhecimento de operadores lógicos == e &&, além do conhecimento em condicionais if e else if e das habilidades de resolver logicamente um problema para listar os casos em que um movimento é inválido.

- `int aleatorio()`

Usamos o conhecimento da biblioteca time para poder gerar numeros randômicos, aprendido na aula.

- `void imprime_menu()`, `void creditos()` e `void regras()`

Usamos primariamente o conhecimento sobre printf, aprendido nas primeiras aulas do curso, para imprimir uma informação na tela.

- `int verifica_fim(char matriz[LINHA][COLUNA])`

Usamos o conhecimento de operadores lógicos == e && para determinar se haviam mais movimentos válidos para cada posição de cada peça no tabuleiro em relação ao espaço vazio, pois se uma peça está no canto inferior direito, por exemplo, e a peça vazia está no canto superior direito, não há movimentos para essa peça.

Assim, fazendo essa verificação para cada caso, podemos ver se algum jogador não possui movimentos, retornando um valor específico para cada caso (1,-1 e 0, caso ninguém perdeu).

- `void reseta_tabuleiro(char matriz[LINHA][COLUNA])`

Usamos o conhecimento de atribuição entre matrizes para voltar o tabuleiro ao seu estado inicial utilizando for-loops, pois é necessário modificar o valor elemento por elemento para “igualar” uma matriz, no nosso caso.

- `void game_loop(char matriz[LINHA][COLUNA])`

Esse é a função do jogo, antes de tudo, o conhecimento de loops foi extremamente necessário para fazer dois loops relacionados, um da partida e o outro da jogada.

Usamos um do-while que inicia o jogo e continua enquanto os jogadores desejarem continuar jogando. Usamos a função `int aleatorio()` anteriormente definida para determinar qual jogador começa. Depois disso, usamos um while-loop para que, enquanto alguém não ganhe, o jogo continue rodando. Usamos o getchar, e limpeza de buffer para receber a entrada do usuário (para nenhum valor indesejado seja inserido, como aprendido nas aulas), representando o movimento que ele deseja fazer. O movimento é checado em um while-loop.

Enquanto o jogador não inserir uma peça válida, testado pela função `int peca_valida (char peca, int jogador)`, o programa imprime na tela "peça inválida". Caso o usuário esteja fazendo um movimento inválido, testado pela função `void verfica_movimentos (char matriz[LINHA][COLUNA], char peca)` o programa imprime na "tela movimento inválido". Quando o jogador finalmente inserir uma peça válida com movimento válido, o programa chama a função `void movimenta (char matriz[LINHA][COLUNA], char peca)` que movimenta a peça, como explicado anteriormente.

Após isso, o programa confere se há um vencedor através de condicionais if e else-if, e, caso haja ganhador imprime a devida mensagem na tela e incrementa a vitória para o jogador que ganhou.

Perceba que durante a maior parte dessas iterações limpamos a tela com a função `void limpa_tela()` para limpar a tela e deixar o terminal mais limpo e o jogo mais bonito.

Quando finalmente alguém ganhar, o placar dessa pessoa é atualizado e é perguntado aos jogadores se eles desejam continuar. Caso contrário o do_while mencionado mais acima é encerrado e essa função também.

- `int main(void)`

É a que inicia o jogo, nela chamamos a função que imprime o menu e utilizamos um switch-case para chamar as funções correspondentes às apresentadas ao jogador. E é isso basicamente, já que para deixar mais acessível e de fácil manutenção, todo o jogo foi modularizado.

Utilizamos um fgets (Aprendido nas aulas, que é mais seguro que gets) para receber a entrada do jogador e, após isso, usamos a strtol para converter essa entrada para um número (evitando assim um valor indesejado), além do while para deixar reutilizar o menu até que se deseje realmente sair (conhecimento adquirido nas listas passadas ao longo do curso).

Por exemplo, o menu diz: "2.Creditos", caso o jogador insira 2, o switch case entrará no case 2 e chamará a função `creditos()`, e esse é a main!

CONCLUSÃO

Definitivamente, como podemos perceber, todos os conhecimentos do curso foram essenciais para a produção, desenvolvimento e depuração do jogo. Além de um grande conhecimento sobre um jogo de tabuleiro desconhecido por todos os integrantes.

Portanto, queríamos agradecer aos monitores pela ajuda ao longo do curso, e a professora Valéria pelas aulas!