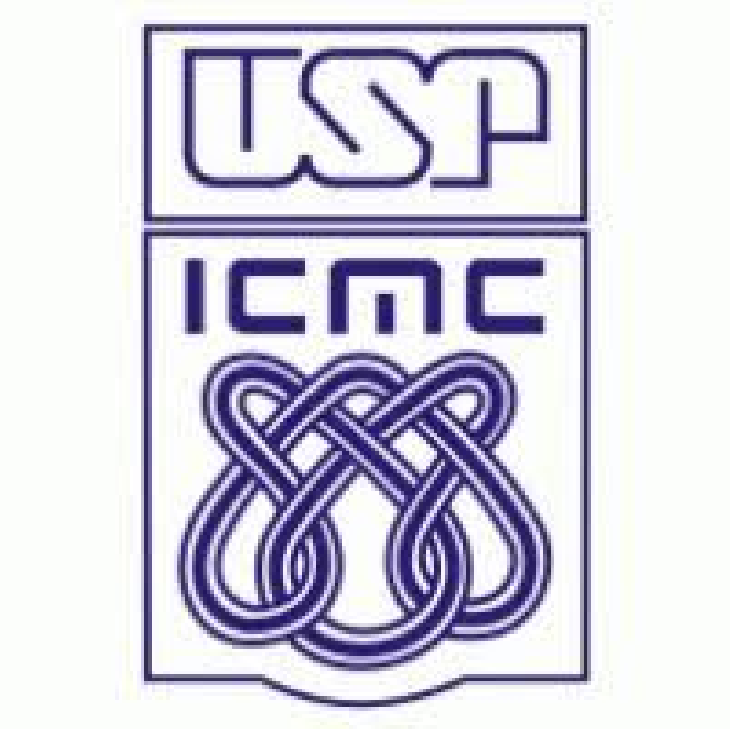
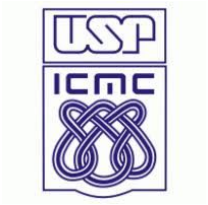


Trabalho 4 - Rummikub

P0 - Thiago Daniel Cagnoni de Pauli - N°USP: 10716629
P1 - Murilo Leone - N°USP: 10748326
P2 - Pedro Ramos Cunha - N°USP: 10892248
P3 - Vitor Rodrigues Leonardi - N°USP: 10716567



Relatório do Trabalho Rummikub

Introdução:

Este Programa executável consiste na implementação em linguagem C de um protótipo do jogo Rummikub comercializado no Brasil pela Grow Jogos e Brinquedos Ltda. Os membros de desse grupo são Thiago Daniel Cagnoni de Pauli (10716629), Murilo Leone (10748326), Pedro Ramos Cunha (10892248) e Vitor Rodrigues Leonardi (10716567).

As funcionalidades contidas neste programa são:

- Escolher o número de jogadores
- O jogador pode manipular o tabuleiro e o próprio.
- Há dois tipos de combinações possíveis de manipulação e descarte no jogo: os grupos e as sequências.

  Grupos (trincas e quadras) são conjuntos de 3 ou 4 peças com o mesmo número e necessariamente com cores diferentes.

  Sequências são conjuntos de 3 a 13 peças de mesma cor, com números em sequência.

- O coringa pode ser usado para completar qualquer combinação, passando a valer o número da peça que ele está a substituir, no grupo ou na sequência.
- O jogo retorna uma mensagem de jogada invalida caso o valor jogado a mesa seja menor que 30 pontos.
- Sempre que um jogador não tiver nada para jogar, ou não puder fazer um mínimo de 30 pontos na sua primeira jogada, ele deverá "comprar", ou seja, escolher mais uma peça do "monte" e adicioná-la às suas peças.
- Durante a vez de cada jogador, as peças sobre a mesa podem ser manipuladas à vontade, formando novas combinações. As únicas obrigações do jogador são: (a) ao final de sua jogada, todas as peças sobre a mesa devem fazer parte de combinações válidas; (b) ele deve ter jogado pelo menos uma peça nova sobre a mesa; (c) se esta for a sua primeira jogada válida, as peças jogadas por ele devem somar pelo menos 30 pontos.

  Exemplos de manipulações possíveis:

Cortar uma sequência: o jogador pode retirar a peça inicial ou final de uma sequência para usá-la em outra combinação, desde que a sequência permaneça com pelo menos três peças.

Cortar uma quadra: o jogador pode retirar uma das peças de uma quadra (que se torna, então, uma trinca) para usá-la em outra combinação. O mesmo não pode ser feito em uma trinca, já que uma combinação de apenas duas peças não seria válida.

Deslocar uma sequência de três: ao se colocar uma peça na ponta (início ou final) de uma sequência de três peças, esta passa a ter quatro peças, e portanto a outra ponta pode ser retirada para uso em outra combinação.

Substituir numa trinca: numa trinca sobre a mesa, o jogador pode adicionar a peça de mesmo número e da cor que falta, transformando-a uma quadra, e portanto podendo retirar uma das outras peças da trinca para usá-la em outra combinação.

Dividir uma sequência: um jogador pode dividir uma sequência longa e colocar as peças correspondentes no meio, desde que as sequências resultantes tenham, cada uma, um mínimo de três peças. Por exemplo, se há na mesa uma sequência azul de 6 a 10 e o jogador tem na mão um 8 azul, ele pode colocar a sua peça no meio e formar as sequências 6-7-8 e 8-9-10.

Substituir um coringa: se o jogador possui a peça que substituiu um coringa em uma combinação sobre a mesa, ele pode trocá-la, podendo a seguir usar o coringa em qualquer outra combinação.

Descrição do projeto:

O projeto foi desenvolvido com a ajuda do programa Sublime text 3, com o GitHubDesktop na versão 1.2.3 e GitKraken na versão 3.6.3.

No desenvolvimento do programa foi utilizado um processador Intel® Core ™ i5-6200 com memória RAM de 8 GB, no Windows 10 Home de processador com base em 64 bits (x64).

Na pasta do trabalho, contém os seguintes arquivos:

- Arquivos .c para serem compilados (main.c, Rummikub.c, tabuleiro.c e checar.c).
- Cabeçalho Rummikub.h.
- Executável Rummikub.exe.
- Entradas exemplos entradas_exemplos.txt.
- README.md
- Relatório em pdf. Relatório_T4.pdf

Tutorial:

Compilação:

Utilizando o compilador GCC, a compilação pode ser feita pelo comando no terminal:

```
>gcc main.c Rummikub.c tabuleiro.c checar.c -o Rummikub
```

Programa:

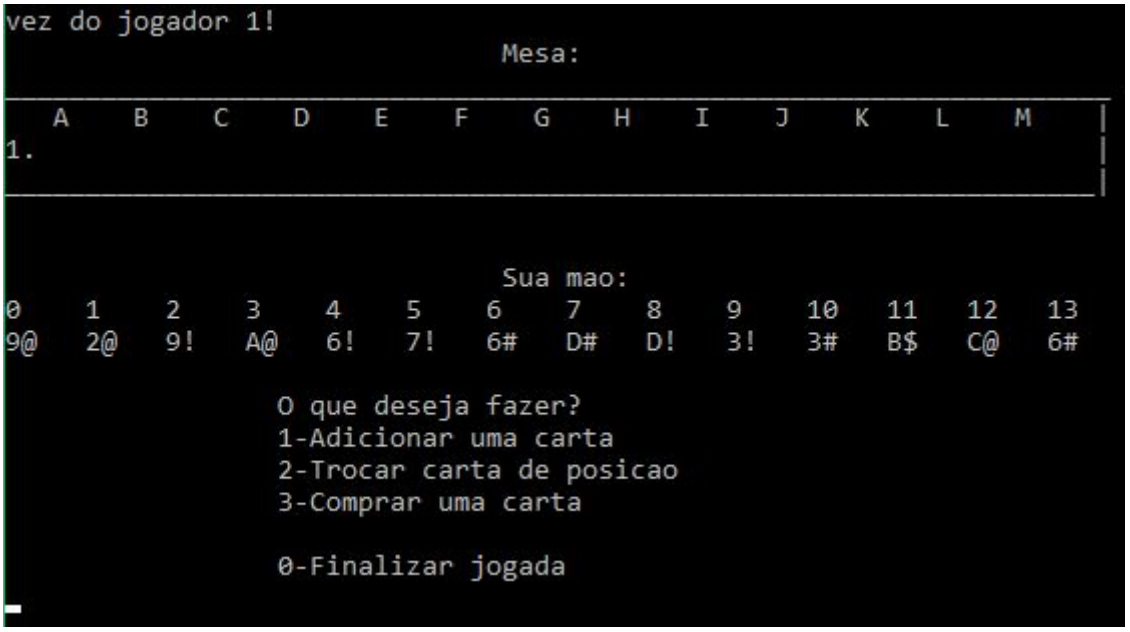
Primeiramente, é necessário definir o número de jogadores:

```
                        ----------------------------
                        |   Bem Vindo ao Rummikub!  |
                        ----------------------------
Numero de jogadores(1 - 5):3
```

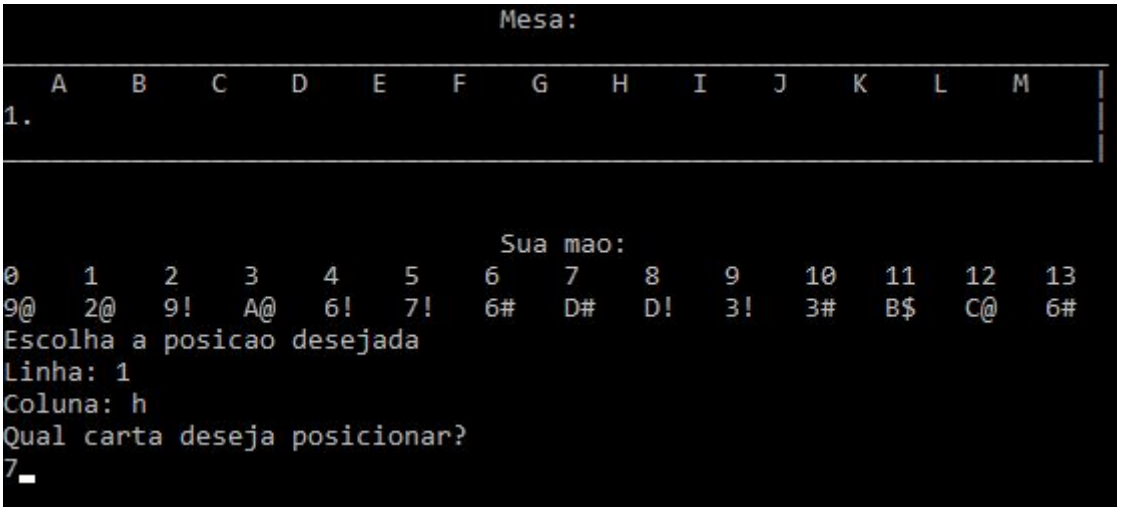
Após isso, o usuário define a maneira como será distribuído as cartas do baralho, sendo as opções de maneira aleatória ou com o arquivo de texto:

```
Como deseja distribuir o baralho?(1-aleatorio/2-Arquivo de texto)
1
```

Após esses passos será exibido na tela a mesa com suas respectivas linhas e colunas, a mão do jogador , e também as opções de comandos:

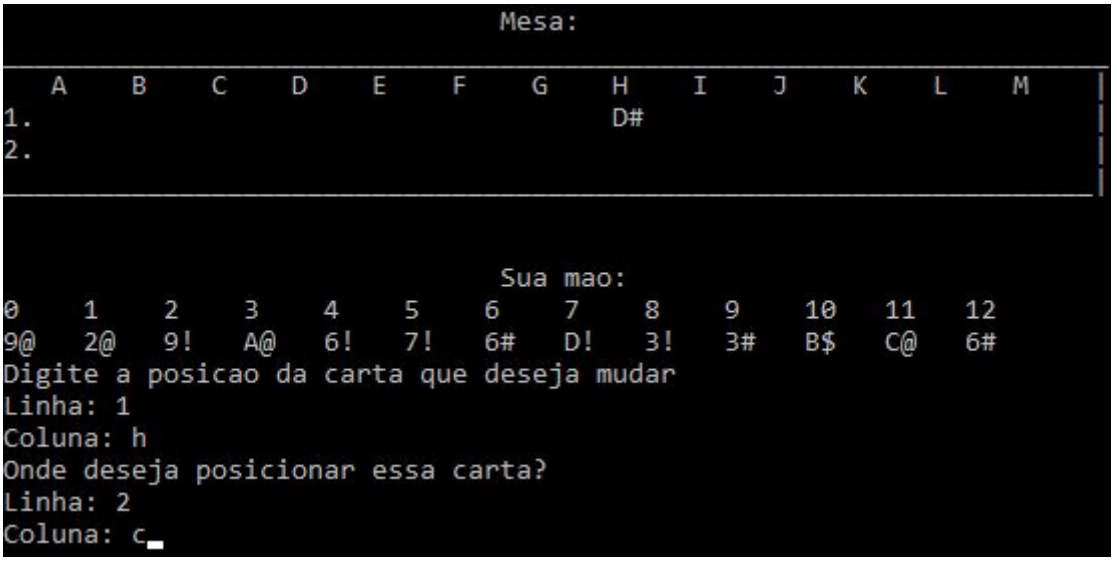


Selecionando a opção “1-Adicionar uma carta” será pedido ao jogador para informar a posição que receberá a carta e qual carta ele deseja adicionar:



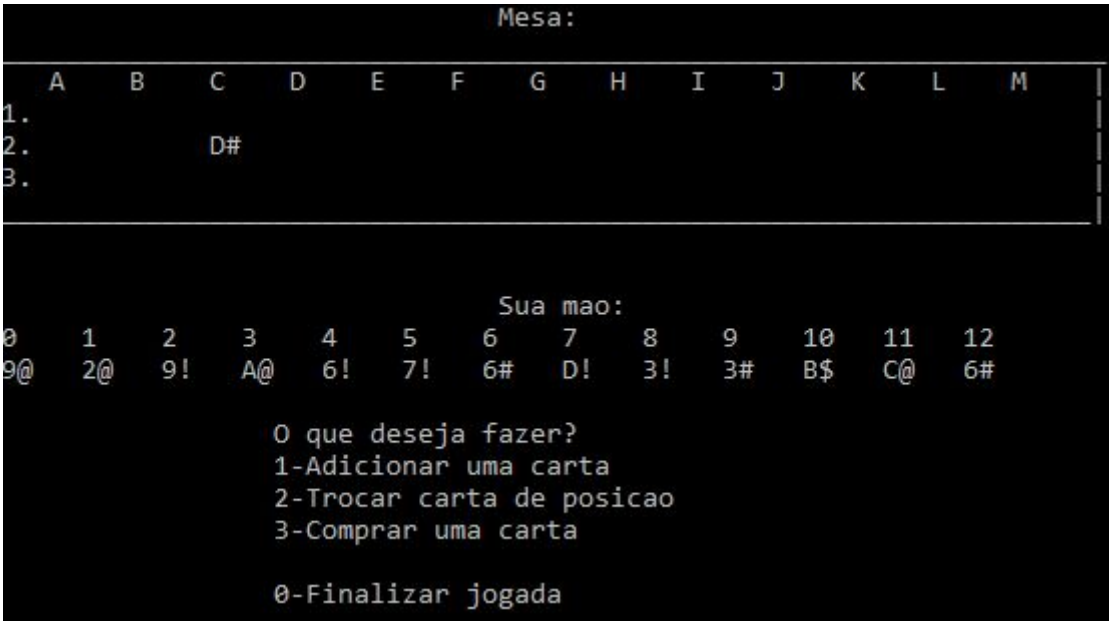
Com isso a carta será retirada da mão do jogador e será colocada na mesa, como mostrado a seguir:

Na opção “2-Trocar carta de posicao”, será pedido para o jogador informar a posição da carta que ele deseja mudar e qual será a posição final da carta:

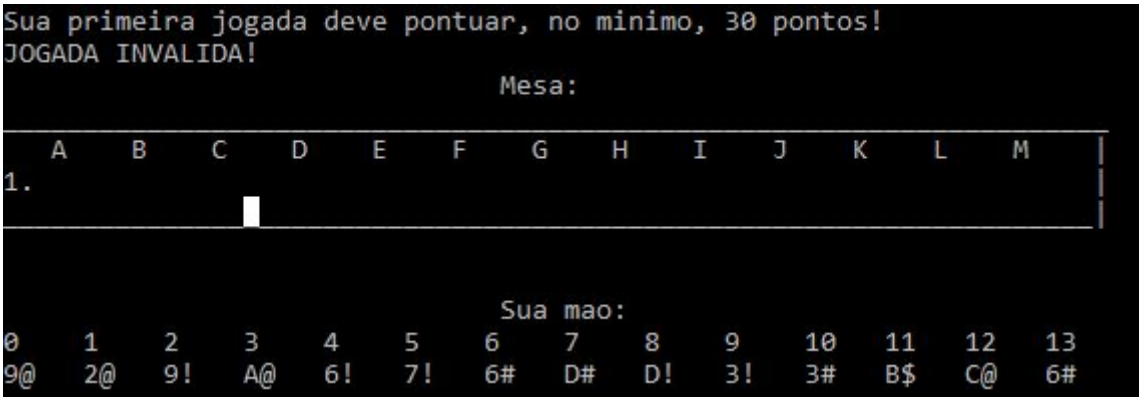


‘

Após esses passos, a mesa será exibida com a mudanças feitas:



Como essa foi a primeira jogada do usuário e ela não somou 30 nos números das cartas jogadas por ele, ao finalizar a jogada , a vez desse jogador será repetida, reiniciando a jogada por inteiro:



Ao comprar uma carta, a vez do jogador passará e na sua próxima jogada, terá uma carta a mais em sua mão:

Bugs e Limitações

Alguns bugs encontrados e corrigidos no programa foram:

- erros de ponteiros.
- jogadas impossíveis sendo válidas
- jogadas inválidas lidas como válidas

Já as limitações em relação ao jogo principal, é que não é possível flexibilizar as regras.
Todos os bugs foram sendo corrigidos conforme foram sendo encontrados, porém, devido à ampla possibilidade de jogadas, é bem provável que seja encontrado um novo erro.