UNIVERSIDADE DE SÃO PAULO

André Bermudes Viana - 10684580

TRABALHO 4 - DE SSC0600 EM CONJUNTO COM SSC0601

Introdução a Ciência da Computação I

Laboratório de Introdução a Ciência da Computação I

São Carlos

2018

1. INTRODUÇÃO

Grupo do Trabalho:

- (P=0) André Bermudes Viana (10684580)

O objetivo do trabalho é implementar um programa utilizando a linguagem C para se jogar Rummikub, o programa permite até 5 jogadores simultaneamente. Cada jogar recebe 14 peças inicialmente, o objetivo do jogo é conseguir colocar todas as peças da sua mão na mesa.

As peças possuem valores que variam de 1-D (hexadecimal), podem ser de 4 “naipes” diferentes (!,@,#,$), além disso o jogo conta com 2 coringas (**). No total o jogo possui 106 peças.

O programa foi feito de maneira que permite a livre manipulação das peças da mesa e da mão do jogador durante sua jogada, porém, o jogador só poderá terminar sua jogada caso o estado atual do jogo seja válido, o jogador a qualquer momento pode também restaurar a jogada ao seu estado inicial.

O programa pode sortear as cartas aleatoriamente entre os jogadores, ou pode ler o “monte de cartas” de um arquivo “baralho.txt” que deve ser colocado na pasta onde se encontra o programa.

2. DESCRIÇÃO DO PROJETO

O programa foi desenvolvido em Windows 10 – 64 bits, utilizando o Codeblocks 17.12, versão 32 bits**,** foi compilado utilizando o GCC versão 6.3.0 (MinGW.org GCC-6.3.0-1), versão win32.

Os códigos fontes utilizados foram o “main.c” (contém a função principal) e “funcoes.c” (contém as sub-rotinas criadas), além dos *headers* “funcoes.h” e “struct.h”.

O programa pode ler um arquivo “baralho.txt” que deve ficar no diretório do programa, no arquivo *.zip* disponibilizado existe um baralho exemplo.

3. TUTORIAL

3.1 Compilando o Programa

Dentro do arquivo compactado “.zip”, já existe um executável do programa, mas também é possível compilá-lo usando os códigos fonte.

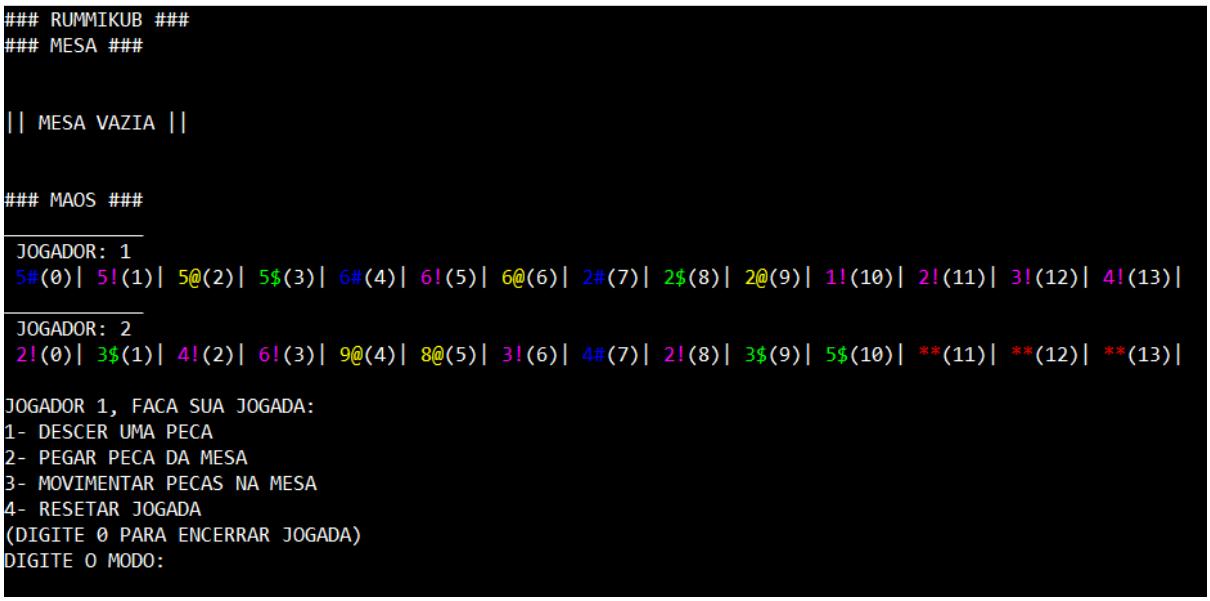
É possível abrir os códigos fonte em programas como Codeblocks, DevC++, Visual Studio, entre outros, e compilar o programa através deles.

Também é possível compilá-lo através do gcc, primeiro instale o gcc (é recomendada a instalação do MinGW), após isso coloque os arquivos do programa na pasta onde está instalado o gcc, abra o prompt de comando e digite o seguinte comando:

gcc -o Romeu_Cubo.exe main.c funcoes.c funcoes.h struct.h

3.2 Utilizando o Programa

Ao executar o programa, ele irá perguntar se deseja ler o arquivo, ou usar um monte aleatório de cartas, além do número de jogadores, após isso, uma tela



semelhante a essa aparecerá:

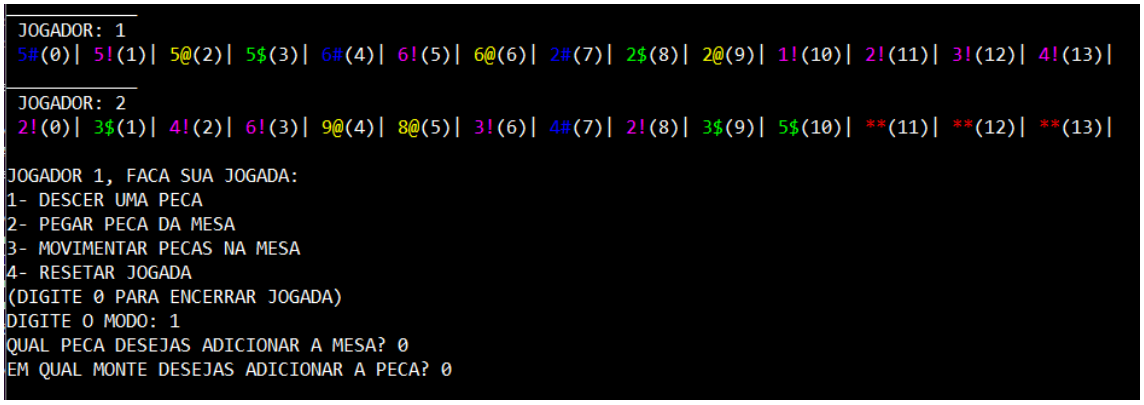
OBS: Nesse exemplo foi utilizado um baralho puramente para teste, por isso existem peças repetidas.

O jogador pode então escolher qual das funções deseja utilizar (digitando o número respectivo), ou pode digitar 0 para encerrar a jogada (porém a jogada somente será finalizada se o estado atual do jogo for válido).

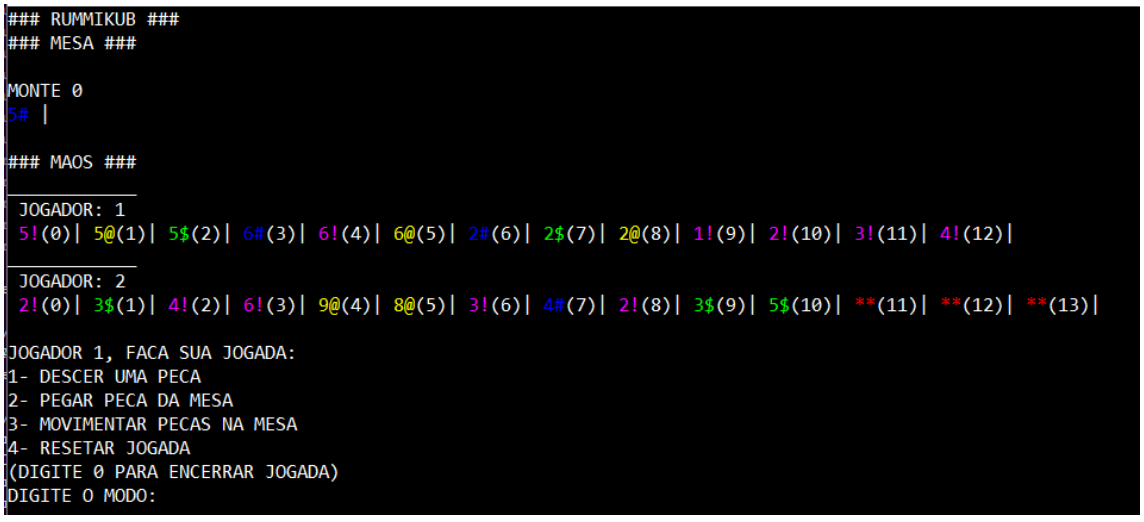
3.2.1 Funções

Segue explicações sobre as 4 funções que o jogador pode escolher:

Função 1 – Descer uma Peça, essa função permite que o jogador coloca uma peça da sua mão em algum dos montes da mesa (ou em um novo monte).



No exemplo acima o jogador 1 quer colocar a peça 5# (0) no monte 0, como essa é a primeira jogada, o monte 0 não existe, e será criado:



OBS: O jogador só poderá criar um novo monte caso ele seja o próximo monte, por exemplo, no caso da última imagem, o jogador pode colocar mais uma carta no monte 0 ou criar o monte 1, mas não pode criar o monte 2.

Função 2 – Pegar Peça da Mesa, essa função permite que o jogador retire uma peça da mesa e coloque a na sua mão. Essa jogada não é permitida no jogo, mas ela foi adicionada no programa para que o jogador possa “desfazer” alguma jogada errada. O programa não permite que o jogador termine sua jogada com alguma peça da mesa em sua mão.

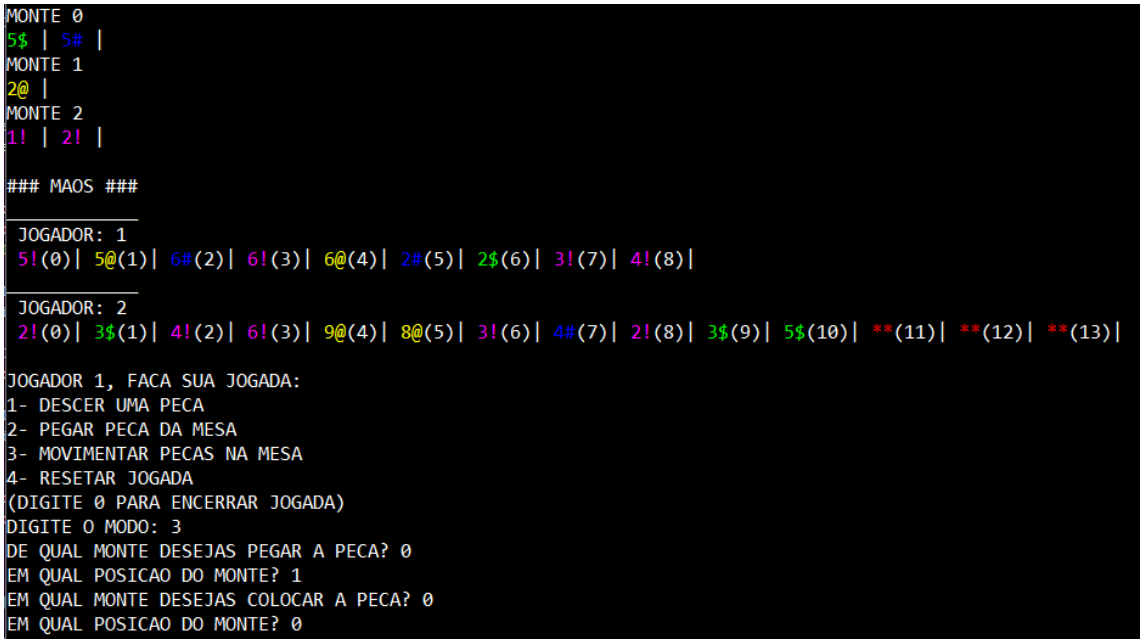
```
### MESA ###

MONTE 0
5# | 5! | 5$ |
MONTE 1
2@ |
MONTE 2
1! | 2! |

### MAOS ###
_____________
 JOGADOR: 1
 5@(0)| 6#(1)| 6!(2)| 6@(3)| 2#(4)| 2$(5)| 3!(6)| 4!(7)|
_____________
 JOGADOR: 2
 2!(0)| 3$(1)| 4!(2)| 6!(3)| 9@(4)| 8@(5)| 3!(6)| 4#(7)| 2!(8)| 3$(9)| 5$(10)| **(11)| **(12)| **(13)|

JOGADOR 1, FACA SUA JOGADA:
1- DESCER UMA PECA
2- PEGAR PECA DA MESA
3- MOVIMENTAR PECAS NA MESA
4- RESETAR JOGADA
(DIGITE 0 PARA ENCERRAR JOGADA)
DIGITE O MODO: 2
DE QUAL MONTE DESEJAS PEGAR A PECA? 0
EM QUAL POSICAO DO MONTE? 1
```

No caso acima o jogador 1 quer pegar a peça 5! que está no monte 0.

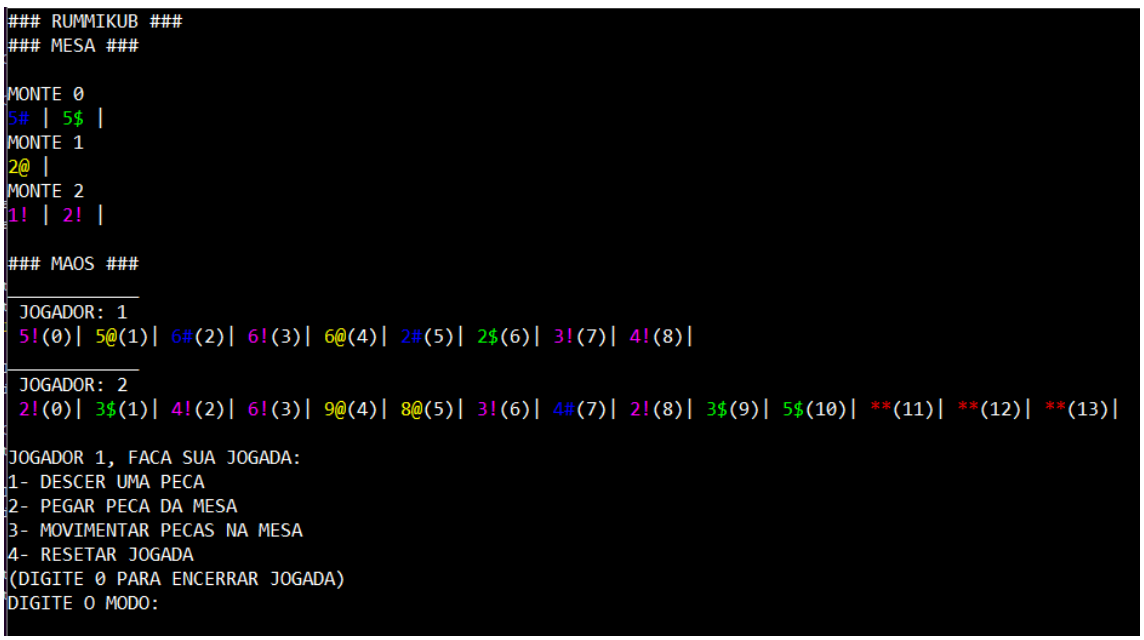
```
MONTE 0
5# | 5$ |
MONTE 1
2@ |
MONTE 2
1! | 2! |

### MAOS ###
_____________
 JOGADOR: 1
 5@(0)| 6#(1)| 6!(2)| 6@(3)| 2#(4)| 2$(5)| 3!(6)| 4!(7)| 5!(8)|
_____________
 JOGADOR: 2
 2!(0)| 3$(1)| 4!(2)| 6!(3)| 9@(4)| 8@(5)| 3!(6)| 4#(7)| 2!(8)| 3$(9)| 5$(10)| **(11)| **(12)| **(13)|

JOGADOR 1, FACA SUA JOGADA:
1- DESCER UMA PECA
2- PEGAR PECA DA MESA
3- MOVIMENTAR PECAS NA MESA
4- RESETAR JOGADA
(DIGITE 0 PARA ENCERRAR JOGADA)
DIGITE O MODO:
```

Tela após a jogada.

Função 3 – Movimentar Peças da Mesa, essa função permite que o jogador altere a posição de uma peça na mesa. O jogador deve informar qual o monte e posição da carta que deseja mudar de posição, além do novo monte e da nova posição. (O jogador pode também mudar duas peças de posição dentro de um mesmo monte)



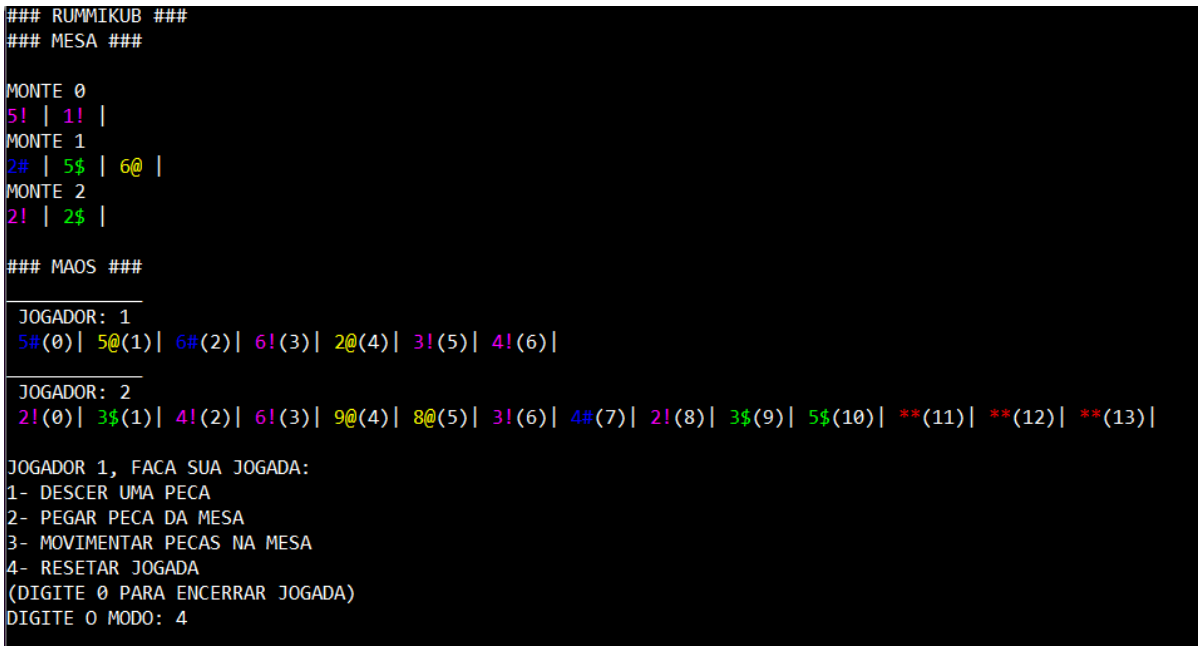
No exemplo, o jogador 1 quer inverter as posições das duas peças no monte 0.

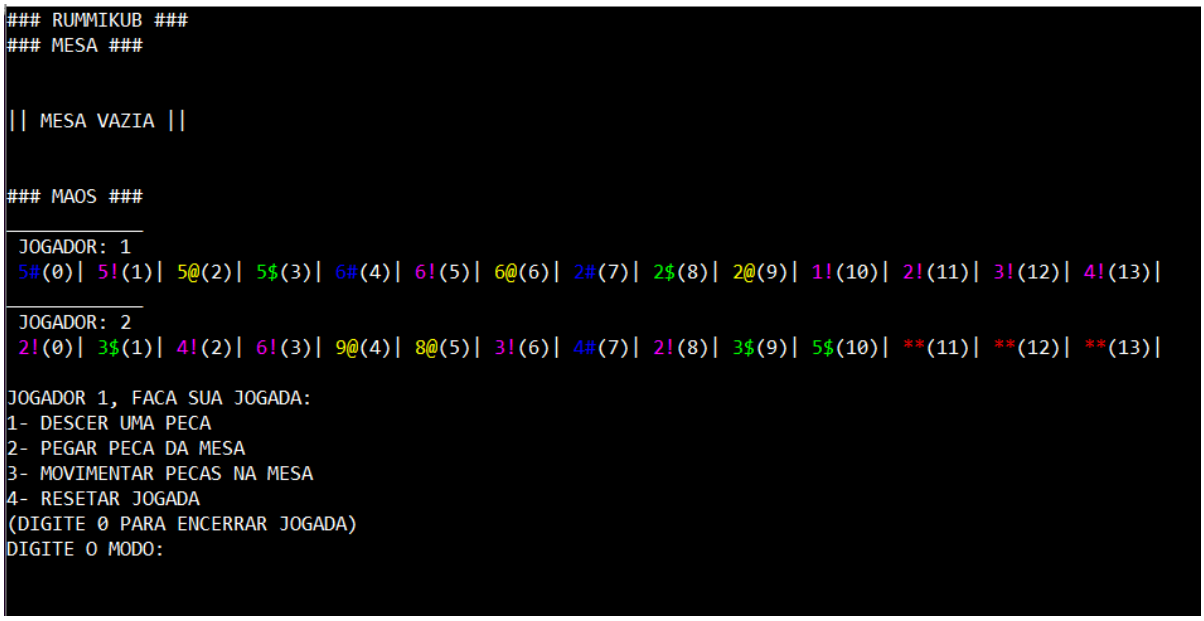


Tela após a jogada.

Função 4 – Resetar Jogada, essa função permite que o jogador desfaça todas as mudanças que fez durante sua jogada, portanto a mesa e a mão do jogador retornarão ao estado antes do jogador começar sua jogada.

Segue imagens do antes e depois da utilização da função:





4. OUTRAS INFORMAÇÕES

4.1 Limitações e Bugs

Na função “Movimentar Peças na Mesa” não existe verificações sobre a posição destino ou monte destino da peça, caso o usuário não digite entradas corretas, a função pode não funcionar como o esperado.

O programa foi feito para Windows, não é recomendado tentar compila-lo no Linux ou outro sistema operacional.

4.2 Tutorial Rummikub

Esse link possui um tutorial sobre o funcionamento do Rummikub, e suas regras.

https://pt.wikipedia.org/wiki/Rummikub