



UNIVERSIDADE  
FEDERAL DO  
RIO DE JANEIRO  

---

UFRJ

Gabriele Jandres Cavalcanti - 119159948

Maria Carolina Esteves de Azevedo - 119146694

Victor Wohlers Cardoso - 119157174

*Relatório do Trabalho - Simulação  
de jogos de tabuleiro  
Resta um*

RIO DE JANEIRO

2019

GABRIELE JANDRES CAVALCANTI  
MARIA CAROLINA ESTEVES DE AZEVEDO  
VICTOR WOHLERS CARDOSO

Trabalho de Computação I apresentado ao curso Ciência da Computação, da  
Universidade Federal do Rio de Janeiro, sob orientação da professora Valéria Menezes  
Bastos.

RIO DE JANEIRO

2019

## Sumário

---

Objetivo do trabalho .....	4
Conhecimentos aplicados .....	4
Descrição do código .....	5
Regras .....	10
Problemas enfrentados .....	13
Saídas .....	14
Bibliografia .....	18

## OBJETIVO DO TRABALHO

O objetivo do trabalho desenvolvido pelo grupo foi, ao desenvolver o jogo RESTA 1, colocar em prática os conhecimentos adquiridos durante o semestre de 2019/2 no curso de Computação 1, ministrado pela professora Valéria Bastos.

## CONHECIMENTOS APLICADOS

Para desenvolvimento do jogo “RESTA 1”, na linguagem C, fizemos uso de:

- Entrada e saída pelo Console;
- Operadores lógicos;
- Vetores;
- Ponteiros;
- Arquivos;
- Alocação Dinâmica;
- Multiarquivos;
- Comandos de Controle;
- Funções;
- Estruturas;
- Bibliotecas: *stdio.h; stdlib.h; string.h; math.h; ctype.h; time.h; locale.h; windows.h; conio.h* e outras criadas por nós mesmos (*Defs.h* e *Molde.h*)

## DESCRIÇÃO DO CÓDIGO

### MENU:

#### 1) Menu inicial

No menu inicial, proporcionamos ao jogador a opção de escolher entre:

1. Jogar
2. Ver Ranking
3. Sair

Função: *void menu\_inicial (char nome[ ], int \*op);*

Esta função tem como parâmetros um vetor do tipo char, que corresponde ao nome do jogador, e um ponteiro para inteiro \*op que corresponde ao valor escolhido pelo jogador quando este escolhe a opção que quer executar.

#### 2) Menu Secundário

Este menu é utilizado para fornecer ao usuário a opção de escolher entre os possíveis preenchimentos de tabuleiro do jogo, isto é, entre os níveis do jogo.

Função: *int menu (char \*nome);*

Esta função tem como parâmetro um ponteiro para char, que corresponde a string nome, correspondente ao nome do jogador atual. Além disso, essa função retorna um valor inteiro correspondente à opção de tabuleiro escolhida pelo jogador.

1. Tabuleiro Pirâmide;
2. Tabuleiro Flecha;
3. Tabuleiro Tradicional.

### TABULEIRO:

#### 1) Montagem

Para construir o tabuleiro, nós construímos três funções, as quais correspondem às três diferentes formas de preenchimento possíveis:

- Flecha (16 peças);
- Pirâmide (17 peças);

- Tradicional (32 peças).

Em cada uma dessas funções o tabuleiro foi construído utilizando um ponteiro para ponteiro do tipo `char (**tab)` para criar uma matriz de caracteres 0, 1 e \*. Utilizamos a função para alocação dinâmica “*malloc*”, dentro de um loop, para alocar o espaço necessário para a criação do tabuleiro.

Depois é feito outro loop para o preenchimento do tabuleiro, de acordo com o formato escolhido, seguindo o seguinte código:

- ‘1’ – Casa ocupada por uma peça;
- ‘0’ – Casa desocupada;
- ‘\*’ – Casa de delimitação do tabuleiro. É um espaço onde não se podem colocar peças.

Funções: `char **tradicional ( );` `char **pirâmide ( );` `char **flecha ( );`

## 2) Impressão:

Para a impressão do tabuleiro na tela criamos uma função que recebe um ponteiro para ponteiro (`**tab`).

Esta função realiza um loop para imprimir os índices que indicam as coordenadas das peças, as quais auxiliam o jogador a se localizar no tabuleiro, e realiza outro loop para imprimir a matriz correspondente ao modelo de tabuleiro escolhido.

Ela é utilizada para todos os tipos de tabuleiro, pois o que determina o formato será exatamente o ponteiro que é recebido como argumento.

Função: `void imprime_tabuleiro (char **tab);`

## EXECUÇÃO DO JOGO:

### 1) Verificação de movimentos

Fizemos uma função que recebe como parâmetros um ponteiro para ponteiro para o tipo `char` e quatro variáveis do tipo inteiro, as quais representam as coordenadas (l – linha & c – coluna) da peça:

- l1 & c1 para origem da peça
- l2 & c2 para destino da peça

Para organizar melhor o código, fornecemos a essa função três possíveis retornos:

- *return* 0: Movimento inválido;
- *return* 1: Movimento válido na vertical;
- *return* 2: Movimento válido na horizontal.

Dividimos desta forma porque para um movimento na vertical, analisamos a distância entre as linhas para realizar a validação. Já no movimento horizontal analisamos a distância entre as colunas.

Com isso em mente, as validações feitas foram as seguintes:

- a) Quando uma peça se move somente uma coordenada é modificada: a linha, em movimentos verticais, ou a coluna, em movimentos horizontais.
- b) A casa entre a casa de origem e a casa de destino deve estar preenchida (ou seja, deve ser igual a '1').
- c) A casa de origem deve ter uma peça (ou seja, deve ser igual a '1').
- d) A casa de destino deve estar vazia (ou seja, deve ser igual a '0').
- e) As coordenadas fornecidas devem respeitar o limite do tabuleiro
- f) As distâncias entre a casa de origem e a casa de destino devem respeitar a seguinte regra:

- $D = |c1 - c2| = 2$ (Movimento Horizontal)

Ou

- $D = |l1 - l2| = 2$ (Movimento vertical)

Função: *int verifica\_movimento (char \*\*p, int l1, int c1, int l2, int c2);*

## 2) Realização do movimento

Fizemos uma função que recebe como parâmetros um ponteiro para ponteiro para o tipo char, quatro variáveis do tipo inteiro, as quais representam as coordenadas (l1, c1, l2, c2) da peça, e uma variável do tipo inteiro que fornece o retorno da função *verifica\_movimento*.

Para esta função utilizamos um switch para separar os casos de movimento de acordo com o valor recebido de *verifica\_movimento*:

- Case 0: Vai mostrar ao jogador uma mensagem informando que as coordenadas fornecidas resultam em um movimento inválido
- Case 1: Realiza o movimento na vertical

- Case 2: Realiza o movimento na horizontal

Quando o movimento for válido, ele é realizado da seguinte forma:

- O conteúdo da casa de origem recebe '0';
- O conteúdo da casa intermediária recebe '0';
- O conteúdo da casa destino recebe '1'.

Função: *void realiza\_movimento (char \*\*p, int l1, int c1, int l2, int c2, int validacao);*

### 3) Verificação de término do jogo

Essa função recebe como parâmetro um ponteiro para ponteiro para o tipo *char (\*\*p)* e retorna um inteiro, de acordo com o seguinte código:

- *return 0*: Não há mais movimentos possíveis;
- *return 1*: Ainda existem movimentos possíveis.

Para esta função utilizamos um laço *for* que percorre todo o tabuleiro a cada jogada com o objetivo de verificar se ainda existem jogadas disponíveis. Dentro desse *for*, utilizamos a função *verifica\_movimento* que é quem efetivamente realiza tal verificação.

Se o retorno dessa função for '0', o jogo termina.

Função: *int verifica\_fim (char \*\*p);*

### 4) Término do jogo

Caso o jogador ganhe o jogo, abrimos o arquivo "pontuações.txt" em modo "a" e em seguida gravamos os seguintes dados: nome do jogador, tempo que demorou para concluir o jogo e um inteiro que indica qual o nível (qual o tabuleiro) escolhido. Esses dados foram previamente armazenados na estrutura Jogador.

Caso o jogador perca, uma mensagem "Infelizmente, não há mais movimentos. Você quer tentar novamente? (S ou N)" é exibida e o jogador escolhe se quer continuar ou não.

#### RANKING:

Como destacado anteriormente, o usuário tem a opção de escolher "Ver o ranking" no menu inicial. Ao selecionar a opção 2, o comando switch entra em seu caso 2 e então abrimos o arquivo chamado "pontuacoes.txt", que armazena os nomes e o tempo para conclusão de jogo



dos usuários que ganharam alguma partida. Abrimos o arquivo em modo de leitura e lemos cada par (nome, tempo) do arquivo e armazenamos na área apontada por *ranking*, que é um ponteiro para um vetor de estruturas do tipo Jogador. Em seguida, se houverem dados, chamamos a função **mostra\_ranking**, que usa um algoritmo *bubble sort* para ordenar o vetor de estruturas e mostrar até as 5 maiores, se houverem. A função recebe como parâmetros o ponteiro para o vetor de estruturas e uma quantidade n que é a quantidade de elementos armazenados no vetor.

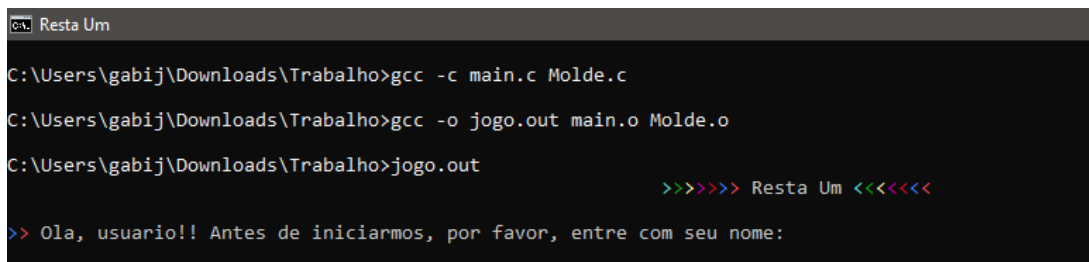
Função: ***void mostra\_ranking(Jogador \*vetor, int n);***

## MULTIARQUIVOS

Em último lugar, mas não menos importante, temos que destacar que é muito difícil criar uma aplicação grande num único código fonte. Por isso, precisamos nos organizar para facilitar o desenvolvimento. E fizemos isso através do uso de módulos, que são arquivos com código fonte da implementação. Em nosso trabalho, usamos os módulos:

- ***main.c*** – arquivo principal
- ***Molde.c*** – arquivo com as definições de funções e de variáveis globais e locais
- ***Molde.h*** – arquivo com alusões de variáveis globais e de funções
- ***Defs.h*** – arquivo com definições de macros e tipos

Para fazer a ligação entre os arquivos, incluímos os cabeçalhos na forma *include "myheader.h"*, onde eram necessários. E, por fim, na linha de comando do terminal, usamos a seguinte sequência de comandos para compilar e em seguida ligar os objetos gerados após a compilação dos arquivos fonte:



```
C:\Users\gabij\Downloads\Trabalho>gcc -c main.c Molde.c
C:\Users\gabij\Downloads\Trabalho>gcc -o jogo.out main.o Molde.o
C:\Users\gabij\Downloads\Trabalho>jogo.out
>>>>> Resto Um <<<<<<
>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
```

## REGRAS

O jogo tem como objetivo sobrar uma única peça sobre o tabuleiro, e de preferência que esta peça esteja no centro do tabuleiro.

Para alcançar esse resultado, as seguintes regras são propostas:

- 1) Mova as peças realizando um salto de uma peça sobre outra (como no jogo de damas).
- 2) Para a realização desses saltos deve-se observar o seguinte:
  - Só se pode saltar sobre uma peça imediatamente ao lado da peça selecionada para realizar o movimento e a casa a ser ocupada após o salto deve estar vazia.
  - Não é permitido pular sobre mais de uma peça de uma só vez
  - Os movimentos só podem ser realizados na horizontal ou na vertical, nenhuma outra direção é permitida.
  - Os saltos podem ser realizados para frente ou para trás
  - Só é permitido dar um salto por jogada
- 3) As peças que foram “puladas” devem ser retiradas do tabuleiro
- 4) O jogo termina quando não houverem mais movimentos possíveis.

## INSTRUÇÕES PARA TESTE DE VITÓRIA DO JOGO

As tabelas abaixo apresentam pares de coordenadas que compõem o caso teste em que o jogador ganha a partida para cada um dos tipos de tabuleiro disponíveis.

### 1) MODO PIRÂMIDE:

L. ORIGEM	C. ORIGEM	L. DESTINO	C. DESTINO
3	2	5	2
3	4	5	4
4	0	4	2
4	6	4	4
5	2	3	2
5	4	3	4
2	2	4	2
2	4	2	2
4	3	4	1
4	1	2	1
2	1	2	3
2	3	4	3
3	5	3	3
4	3	2	3
1	3	3	3

### 2) MODO FLECHA:

L. ORIGEM	C. ORIGEM	L. DESTINO	C. DESTINO
6	4	4	4
4	3	4	5
6	3	4	3
6	2	4	2
4	2	4	4
4	5	4	3
2	4	0	4
0	4	0	2
2	2	2	4
0	2	2	2
2	1	2	3
2	4	2	2
4	3	2	3
2	2	2	4
2	5	2	3
1	3	3	3

3) MODO TRADICIONAL:

L. ORIGEM	C. ORIGEM	L. DESTINO	C. DESTINO
3	5	3	3
1	4	3	4
2	6	2	4
4	6	2	6
2	3	2	5
2	6	2	4
2	1	2	3
0	2	2	2
0	4	0	2
3	2	1	2
0	2	2	2
5	2	3	2
4	0	4	2
2	0	4	0
4	3	4	1
4	0	4	2
4	5	4	3
6	4	4	4
6	2	6	4
3	4	5	4
6	4	4	4
3	2	1	2
1	2	1	4
1	4	3	4
3	4	5	4
5	4	5	2
5	2	3	2
3	3	1	3
3	1	3	3
4	3	2	3
1	3	3	3

## PROBLEMAS ENFRENTADOS

Enfrentamos diversos problemas para a realização do trabalho, mas o principal enfrentado pelo grupo foi em relação à criação do ponteiro para ponteiro para char, que é o ponteiro cuja área apontada é o tabuleiro. A priori, criamos a matriz de char “na mão”, no entanto, ao criar um ponteiro e fazê-lo apontar para tal matriz, estávamos nos deparando com um erro, que posteriormente, descobrimos ser devido ao fato de que a área onde a matriz estava alocada (por ter sido criada dentro de uma função), foi “desalocada” logo assim que o bloco da função foi executado, já que os dados foram colocados na área de pilha e quando a função voltou a seu ponto de retorno, os dados foram desempilhados, e portanto, perdidos. Por isso, criamos o tabuleiro usando alocação dinâmica, assim a área que precisávamos ficaria disponível na “*área de heap*” até que déssemos *free* no ponteiro.

Ademais, enfrentamos problemas para manipular estruturas e arquivos, em princípio, para conseguirmos fazer a função “*mostra\_ranking*”. Ao tentar armazenar os valores da estrutura no arquivo, o programa estava simplesmente parando toda a execução. Porém, depois de algum tempo de pesquisa e dedicação ao projeto, conseguimos finalizar tal funcionalidade.

# SAÍDAS

## Opção 1 do menu - Jogar:

→ **Caso 1:** o jogador perdeu - não existiam mais movimentos possíveis.

```
Resto Um
>>>>> Resto Um <<<<<<

>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
Valeria

>> Bem-vindo, Valeria! O que voce deseja fazer?

    1 - Jogar
    2 - Consultar ranking
    3 - Sair

1

>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:

    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional

3

  0 1 2 3 4 5 6
0 * * 1 1 1 * *
1 * * 1 1 1 * *
2 1 1 1 1 1 1
3 1 1 1 0 1 1
4 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *

Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:
3
Digite a coluna de origem:
5

Digite as coordenadas de onde voce deseja colocar a peca:
Digite a linha de destino:
3
Digite a coluna de destino:
3
```

```
Resto Um

Digite a linha de destino:
3
Digite a coluna de destino:
3

  0 1 2 3 4 5 6
0 * * 1 1 1 * *
1 * * 1 1 1 * *
2 1 1 1 1 1 1
3 1 1 1 1 0 0 1
4 1 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *

Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:
1
Digite a coluna de origem:
4

Digite as coordenadas de onde voce deseja colocar a peca:
Digite a linha de destino:
3
Digite a coluna de destino:
4

  0 1 2 3 4 5 6
0 * * 1 1 1 * *
1 * * 1 1 0 * *
2 1 1 1 1 0 1 1
3 1 1 1 1 1 0 1
4 1 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *

Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:
```

```
Restá Um
2
Digite a coluna de destino:
4

  0 1 2 3 4 5 6
0 * * 1 1 1 * *
1 * * 1 1 0 * *
2 1 1 0 0 1 1 1
3 1 1 1 1 1 0 1
4 1 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *

Digite as coordenadas da peça que você deseja mover:
Digite a linha de origem:
3
Digite a coluna de origem:
3
Digite as coordenadas de onde você deseja colocar a peça:
Digite a linha de destino:
3
Digite a coluna de destino:
1

Movimento invalido. Tente outro par de coordenadas!

  0 1 2 3 4 5 6
0 * * 1 1 1 * *
1 * * 1 1 0 * *
2 1 1 0 0 1 1 1
3 1 1 1 1 1 0 1
4 1 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *

Digite as coordenadas da peça que você deseja mover:
Digite a linha de origem:
```

...

```
Restá Um
Digite as coordenadas de onde você deseja colocar a peça:
Digite a linha de destino:
4
Digite a coluna de destino:
3

  0 1 2 3 4 5 6
0 * * 0 0 0 * *
1 * * 0 0 0 * *
2 1 0 0 0 0 0 0
3 1 0 0 0 0 0 0
4 1 0 0 1 1 0 0
5 * * 1 0 0 * *
6 * * 0 0 1 * *

Digite as coordenadas da peça que você deseja mover:
Digite a linha de origem:
4
Digite a coluna de origem:
3
Digite as coordenadas de onde você deseja colocar a peça:
Digite a linha de destino:
4
Digite a coluna de destino:
5

  0 1 2 3 4 5 6
0 * * 0 0 0 * *
1 * * 0 0 0 * *
2 1 0 0 0 0 0 0
3 1 0 0 0 0 0 0
4 1 0 0 0 0 1 0
5 * * 1 0 0 * *
6 * * 0 0 1 * *

Infelizmente, não há mais movimentos. Você quer tentar novamente? (S ou N)
```

```
cc RestaUm
Digite a coluna de origem:
3
Digite as coordenadas de onde voce deseja colocar a peca:
Digite a linha de destino:
4
Digite a coluna de destino:
5

  0  1  2  3  4  5  6
0  *  *  0  0  0  *  *
1  *  *  0  0  0  *  *
2  1  0  0  0  0  0  0
3  1  0  0  0  0  0  0
4  1  0  0  0  0  1  0
5  *  *  1  0  0  *  *
6  *  *  0  0  1  *  *

Infelizmente, nao ha mais movimentos. Voce quer tentar novamente? (S ou N)
s
>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:

    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional
2

  0  1  2  3  4  5  6
0  *  *  0  1  0  *  *
1  *  *  1  1  1  *  *
2  0  1  1  1  1  1  0
3  0  0  0  1  0  0  0
4  0  0  0  1  0  0  0
5  *  *  1  1  1  *  *
6  *  *  1  1  1  *  *

Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:
```

→ **Caso 2:** o jogador ganhou

```
>>>>>> Resta Um <<<<<<

>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
Joana

>> Bem-vindo, Joana! O que voce deseja fazer?

    1 - Jogar
    2 - Consultar ranking
    3 - Sair
1

>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:

    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional
1

  0  1  2  3  4  5  6
0  *  *  0  0  0  *  *
1  *  *  0  1  0  *  *
2  0  0  1  1  1  0  0
3  0  1  1  1  1  1  0
4  1  1  1  1  1  1  1
5  *  *  0  0  0  *  *
6  *  *  0  0  0  *  *

Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:
5
Digite a coluna de origem:
2

Digite as coordenadas de onde voce deseja colocar a peca:
Digite a linha de destino:
5
Digite a coluna de destino:
2
```

...



```
CU Prompt de Comando
Digite a linha de destino:
2
Digite a coluna de destino:
3
0 1 2 3 4 5 6
0 * * 0 0 0 * *
1 * * 0 1 0 * *
2 0 0 0 1 0 0 0
3 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0
5 * * 0 0 0 * *
6 * * 0 0 0 * *

Digite as coordenadas da peça que voce deseja mover:
Digite a linha de origem:
1
Digite a coluna de origem:
3
Digite as coordenadas de onde voce deseja colocar a peça:
Digite a linha de destino:
3
Digite a coluna de destino:
3
0 1 2 3 4 5 6
0 * * 0 0 0 * *
1 * * 0 0 0 * *
2 0 0 0 0 0 0 0
3 0 0 0 1 0 0 0
4 0 0 0 0 0 0 0
5 * * 0 0 0 * *
6 * * 0 0 0 * *

Parabens Joana , voce ganhou o jogo!
Foram gastos 1.20 minutos para acabar o jogo
```

## Opção 2 do menu – Consultar ranking:

```
>>>>>> Resta Um <<<<<<

>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
la

>> Bem-vindo, la! O que voce deseja fazer?

    1 - Jogar
    2 - Consultar ranking
    3 - Sair
2

Confira o ranking dos 3 melhores jogadores:

1 lugar - Ana com 2.74 minutos - nivel medio
2 lugar - Julia com 2.96 minutos - nivel facil
3 lugar - Joao com 3.15 minutos - nivel dificil
```

## Opções de tabuleiros:

```
c:\users\joaozy\documents\trabalho\jogo\jogo.py >>>>>> Resta Um <<<<<<

>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
Ana

>> Bem-vindo, Ana! O que voce deseja fazer?

    1 - Jogar
    2 - Consultar ranking
    3 - Sair
1

>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:

    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional
1

0 1 2 3 4 5 6
0 * * 0 0 0 * *
1 * * 0 1 0 * *
2 0 0 1 1 1 0 0
3 0 1 1 1 1 1 0
4 1 1 1 1 1 1 1
5 * * 0 0 0 * *
6 * * 0 0 0 * *

Digite as coordenadas da peça que voce deseja mover:
Digite a linha de origem:
```

Tabuleiro piramidal

```

>>>>> Resta Um <<<<<<
>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
Ana
>> Bem-vindo, Ana! O que voce deseja fazer?
    1 - Jogar
    2 - Consultar ranking
    3 - Sair
1
>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:
    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional
2
  0  1  2  3  4  5  6
0 * * 0 1 0 * *
1 * * 1 1 1 * *
2 0 1 1 1 1 1 0
3 0 0 0 1 0 0 0
4 0 0 0 1 0 0 0
5 * * 1 1 1 * *
6 * * 1 1 1 * *
Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:

```

Tabuleiro de flecha

```

>>>>> Resta Um <<<<<<
>> Ola, usuario!! Antes de iniciarmos, por favor, entre com seu nome:
Ana
>> Bem-vindo, Ana! O que voce deseja fazer?
    1 - Jogar
    2 - Consultar ranking
    3 - Sair
1
>> Escolha um dos niveis de dificuldade abaixo para iniciar o jogo:
    1 - Facil - Tabuleiro piramide
    2 - Medio - Tabuleiro flecha
    3 - Dificil - Tabuleiro tradicional
3
  0  1  2  3  4  5  6
0 * * 1 1 1 * *
1 * * 1 1 1 * *
2 1 1 1 1 1 1 1
3 1 1 1 0 1 1 1
4 1 1 1 1 1 1 1
5 * * 1 1 1 * *
6 * * 1 1 1 * *
Digite as coordenadas da peca que voce deseja mover:
Digite a linha de origem:

```

Tabuleiro tradicional

**Observação:** o jogo foi desenvolvido em ambiente Windows, no entanto, as linhas de código exclusivas para o funcionamento nesse sistema operacional estão indicadas no código. Isto é, para rodar no Linux, existem comentários ao lado das bibliotecas que devem ser comentadas para evitar erros ao rodar o jogo no Linux.

## BIBLIOGRAFIA

- <https://www.divertudo.com.br/restaum/restaum.html>
- <http://educarepersone.blogspot.com/2010/11/didatica-da-matematica-jogo-resta-um.html>
- <http://patriciaoliveirapsicopedagoga.blogspot.com/2017/04/o-jogo-resta-um-um-estudo-psicogenetico.html>
- Curso de Linguagem C - Adriano Joaquim de Oliveira Cruz
- <https://www.vivaolinux.com.br/topico/C-C++/Medir-Tempo-de-Execucao>
- <https://www.cprogressivo.net/2013/09/Modulos-Como-criar-um-projeto-no-Code-Blocks-para-Organizar-e-dividir-seu-codigo.html?m=1>
- <https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences>
- [https://en.cppreference.com/w/cpp/chrono/c/CLOCKS\\_PER\\_SEC](https://en.cppreference.com/w/cpp/chrono/c/CLOCKS_PER_SEC)