

Computação I - MAB120

Trabalho Final -

Senha

Desenvolvido por

David Rodrigues Albuquerque - 120047390

Matheus Cardoso de Souza - 120053228

Seth Ribeiro - 118171816

03 de Março de 2021

Sumário

- 1. Objetivo**
- 2. Premissas**
- 3. Regras**
- 4. Técnicas implementadas**
 - 4.1 Ciclo de Desenvolvimento
 - 4.2 Conhecimentos de Computação I
- 5. Saídas da Execução**
- 6. Dificuldades**
- 7. Considerações Finais**
 - 7.1 Conclusão
 - 7.2 Links para disponibilização do jogo
- 8. Bibliografia**

1. Objetivo

O objetivo principal deste trabalho consiste em organizar, projetar e desenvolver um software de computador utilizando a linguagem C para simular um jogo de tabuleiro. O jogo desenvolvido para a elaboração deste foi **Senha**. É um jogo de enigma em que o jogador tenta, através de várias rodadas de tentativas e dicas, adivinhar a senha gerada pelo oponente.

Buscamos nos mantermos atrelados às regras do jogo original e proporcionar uma experiência próxima ao jogo de tabuleiro através do terminal.

São utilizados caracteres coloridos para representar os “pinos” que compõem a senha, as tentativas do jogador e o feedback da tentativa do jogador.

2. Premissas

Pensamos na experiência do usuário e desenvolvemos um terminal onde o jogador pode ver claramente o que está acontecendo.

Começamos pontuando quais funções precisaríamos para fazer o programa inteiro e depois definimos exatamente qual seria o papel de cada um no jogo. Pensamos também na base, que seria começar pelo menu do jogo e continuar o desenvolvimento por ele.

Definimos todas as tecnologias que usaríamos para implementar o jogo antes de começar, assim como organizar o repositório no Git. Definimos tudo que era mais importante para o jogo funcionar e o que poderia ter no jogo se sobrasse tempo para desenvolver.

3. Regras

As regras e instruções do jogo **Senha** podem ser acessadas através do menu da aplicação. Consiste nos seguintes itens:

1. O jogador terá um número limitado de tentativas (**8**) para acertar a senha gerada na partida.
2. Caso acerte a senha, o jogador ganha. Caso acabe o número de tentativas e o jogador não acerte a senha, o jogador perde.
3. Cada senha é gerada aleatoriamente pela aplicação e consiste em um número pré-determinado (**4**) de caracteres representando a inicial de cada cor.
4. Para acertar a senha, o jogador precisa acertar as 4 cores da senha em suas respectivas posições corretamente.

Exemplo de senha: **RYBM**.

-
5. A cada tentativa inserida pelo jogador, ele receberá um feedback da sua senha inserida. Um pino branco representa que um dos pinos de sua senha tem a cor certa e está na posição certa, um pino preto representa que um dos pinos de sua senha tem a cor certa e está na posição errada.

Legenda:

C → Ciano

G → Verde

M → Magenta

R → Vermelho

B → Azul

B → Preto

Y → Amarelo

W → Branco

Guesses → Tentativa do jogador

Hints → Feedback da aplicação sobre a tentativa do jogador.

4. Técnicas Implementadas

4.1. Para auxiliar no desenvolvimento da aplicação, foram adotadas ferramentas auxiliares para ajudar a projetar, organizar e implementar as funcionalidades do jogo. Foi utilizada a plataforma de hospedagem de código e controle de versão Github (<https://github.com>) juntamente com o cliente gráfico GitKraken (<https://www.gitkraken.com>) para disponibilização, visualização, refatoração e versionamento do código desenvolvido. (*figura 1*)

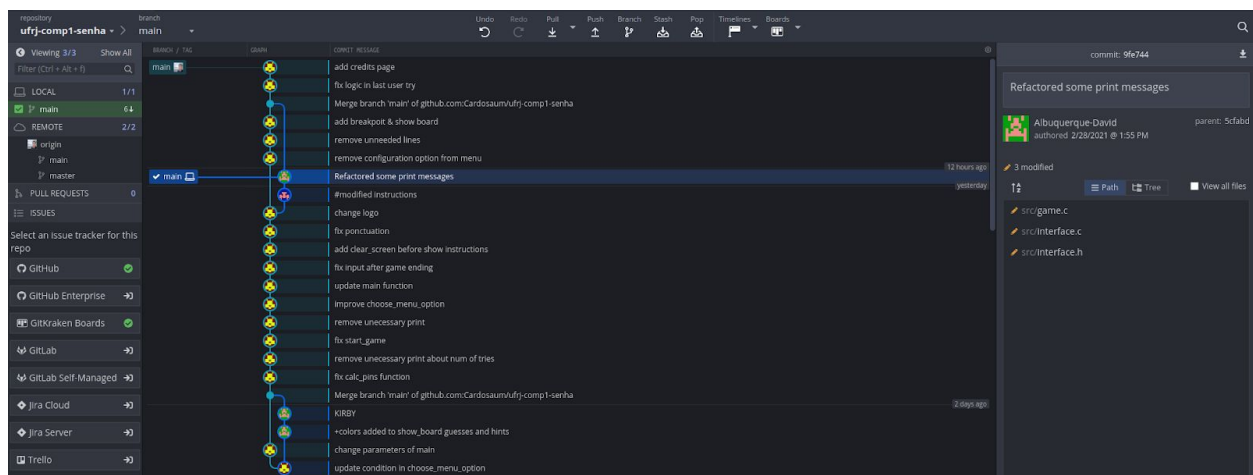


Figura 1: Interface gráfica do software GitKraken com as últimas atualizações no repositório da aplicação.

Para auxiliar no desenvolvimento, também foi implementado um robô automático no serviço de mensageria Telegram. Este robô é responsável por disparar automaticamente para todos os desenvolvedores informações sobre alterações no projeto. (*figura 2*)

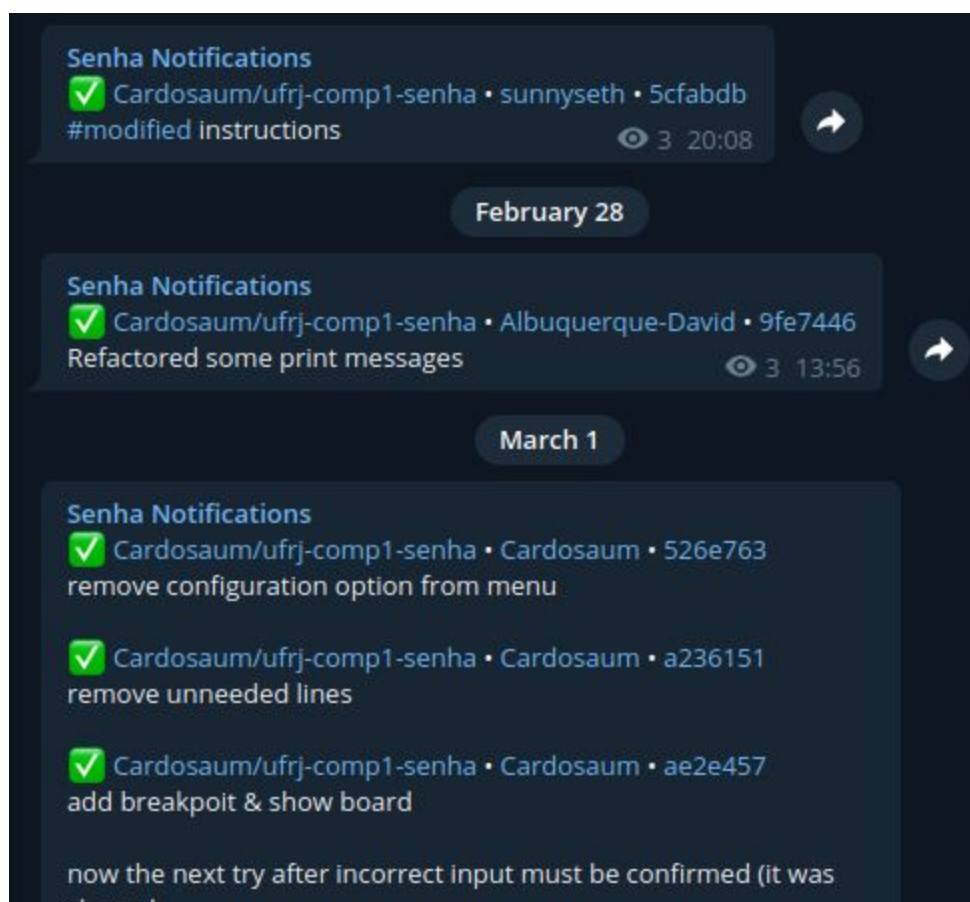


Figura 2: Mensagens recebidas pelo robô no aplicativo Telegram.

Para auxiliar na elaboração e execução das tarefas, foi utilizado também um quadro Kanban do aplicativo Trello (<https://trello.com>) associado ao Github.

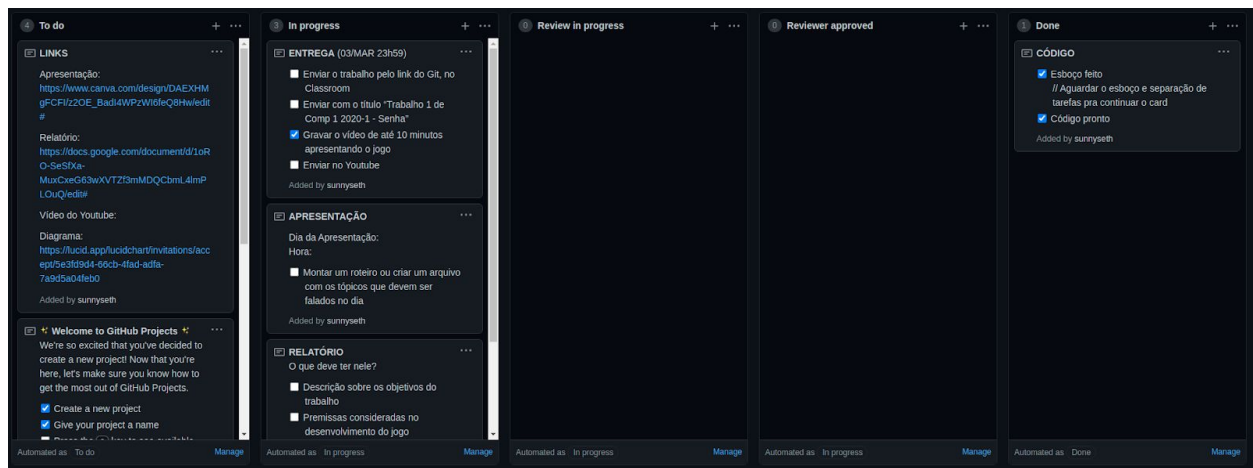


Figura 3: Trello conectado ao Github

4.2. Durante a implementação do jogo em código, foram utilizados conhecimentos aprendidos durante a disciplina de Computação 1 - MAB120 para o desenvolvimento das funcionalidades, regras e saídas.

Foram abordados os seguintes conhecimentos aprendidos durante a disciplina:

- Ponteiros
- Estrutura
- Funções
- Multi Arquivos
- Vetores
- Strings
- I/O
- Comandos de Decisão de Repetição
- Compilação Condicionada
- Macros

Cada um dos tópicos é abordado e detalhado nos arquivos de definição das funções e variáveis da aplicação (arquivos terminados com a extensão .h).

```
typedef struct Password {
    char password[PASSWORD_LENGTH];
} Password;

typedef struct Guess {
    Password player_password;
    int feedback[4];
    bool feedback_given;
} Guess;

typedef struct Board {
    Guess rounds[BOARD_SIZE];
    Password password;
    int tried;
    int won;
} Board;
```

Figura 4: Estruturas utilizadas para representação dos componentes do jogo.

```
• void show_menu();
• int choose_menu_option();
• void show_board(Board board);
• void show_instructions();
• void exit_game();
• void clear_screen();
• void print_logo();
• int calc_pins(char* player_password,
char* game_password, char color);
• Password generate_password();
• bool input_password(Guess *player_guess,
int pos);
• bool check_password(const char
*password);
• bool check_tries();
• void initialize_board(Board *game_board);
• int start_game();
• int finish_game(Board board);
```

Figura 5: Funções utilizadas para a modularização das funcionalidades do jogo.

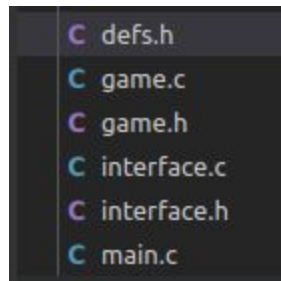


Figura 6: Estrutura de arquivos do projeto.

O desenvolvimento do código e as mensagens exibidas para o jogador foram implementadas em inglês, de modo a utilizar uma linguagem globalmente acessível, tendo em vista que a aplicação ficará disponível em repositório público através da ferramenta Github.

5. Saídas da Execução

Ao inicializar o jogo, o jogador será apresentado ao menu de opções. (*figura 6*)

1. Jogar: Inicia o jogo.

a. O jogador terá exibido o tabuleiro com suas tentativas e feedbacks, e uma linha onde poderá realizar uma tentativa. (*figura 12*)

b. Caso acerte a senha, é exibida a tela de vitória e um novo menu de opções. (*figura 13*)

c. Caso perca, é exibida a tela de derrota e um novo menu de opções. (*figura 14*)

2. Instruções: Na tela de **instruções**, são detalhadas resumidamente as regras do jogos bem como as entradas esperadas pelo usuário e a descrição das saídas do programa. (*figura 15*)

3. Créditos: Na tela de **créditos**, são fornecidas informações dos desenvolvedores da aplicação e seus contatos.
4. Sair: Encerra a execução do programa.

```
MASTERMIND

-----
1. Play
2. Instructions
3. Credits
4. Exit
-----
Which option will you choose? (1/2/3/4)
> █
```

Figura 7: Tela inicial do programa com o menu de opções.

Guesses	Hint
YYYY	----
YRBC	BB--
YMBB	BB--

5 Tries left
Password guess:
> █

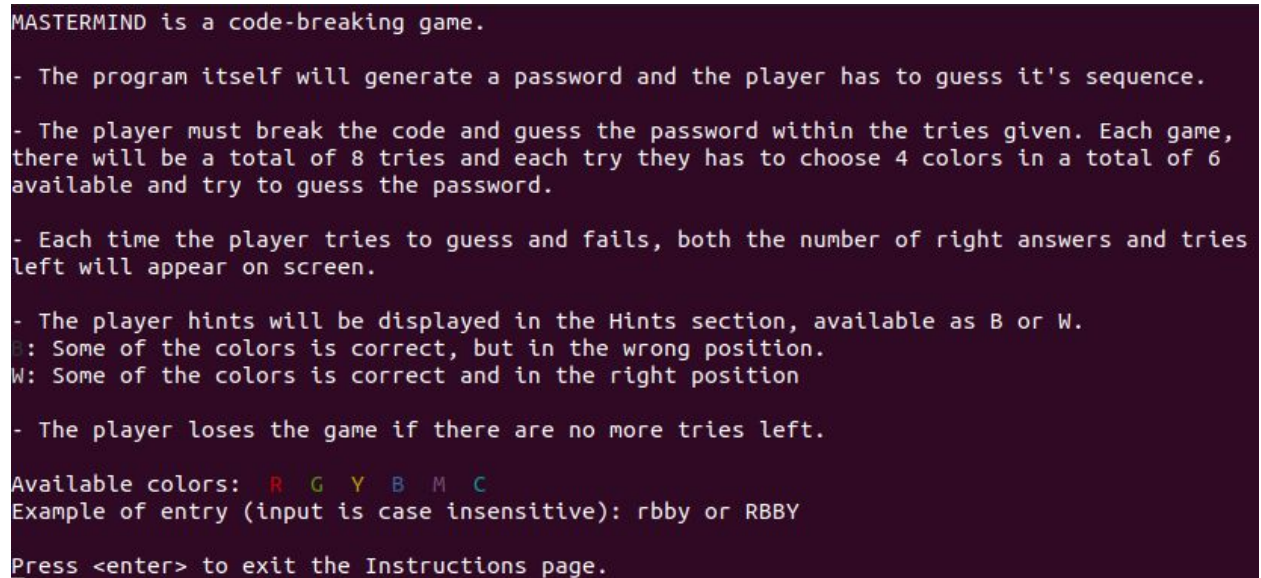
Figura 8: Exibição do tabuleiro e solicitação de entrada do usuário.



Figura 9: Tela de derrota.



Figura 10: Tela de vitória.



MASTERMIND is a code-breaking game.

- The program itself will generate a password and the player has to guess it's sequence.
- The player must break the code and guess the password within the tries given. Each game, there will be a total of 8 tries and each try they has to choose 4 colors in a total of 6 available and try to guess the password.
- Each time the player tries to guess and fails, both the number of right answers and tries left will appear on screen.
- The player hints will be displayed in the Hints section, available as B or W.
 - B: Some of the colors is correct, but in the wrong position.
 - W: Some of the colors is correct and in the right position
- The player loses the game if there are no more tries left.

Available colors: R G Y B M C

Example of entry (input is case insensitive): rbby or RBBY

Press <enter> to exit the Instructions page.

Figura 11: Tela de instruções.



MASTERMIND was developed by:

- David Albuquerque (github.com/Albuquerque-David)
- Matheus Cardoso (github.com/Cardosaum)
- Seth Ribeiro (github.com/sunnyseth)

Figura 12: Tela de créditos.

6. Dificuldades

As principais dificuldades encontradas foram relacionadas à comunicação e ao desenvolvimento de funções para prever o comportamento do usuário. No que tange a comunicação, dado o cenário da pandemia do COVID-19, perde-se a comunicação presencial e há a necessidade de adaptação às ferramentas exclusivamente online para colaboração. Já a respeito das funções, foram encontradas dificuldades na confecção de estratégias que pudessem receber corretamente as entradas do usuário que fossem potencialmente danosas, que pudessem prejudicar o funcionamento do programa (entradas mal formatadas, caracteres especiais, etc.).

7. Considerações Finais

7.1. A elaboração do projeto foi fundamental para associar os conhecimentos aprendidos na disciplina de Computação I em um exemplo prático, onde pudemos ter a liberdade de explorar os recursos da linguagem C para projetar e desenvolver um simulador. O trabalho também demonstrou-se essencial para ajudar na familiarização com ambiente Linux e a adaptar-se às boas práticas de desenvolvimento através do Git.

7.2. Link para repositório Git:
<https://github.com/Cardosaum/ufri-comp1-senha>

Link de acesso ao vídeo de apresentação:
<https://youtu.be/GyF1iDzP29w>

Link de acesso a apresentação em slides:
https://www.canva.com/design/DAEXHMgFCFI/z2OE_Badl4WPzWI6feQ8Hw/edit

8. Bibliografia

Wikipedia, the free encyclopedia, Wikipedia, 2021, Disponível em: [https://en.wikipedia.org/wiki/Mastermind_\(board_game\)](https://en.wikipedia.org/wiki/Mastermind_(board_game)). Acesso em 12/02/2021.

freeCodeCamp, DevDocs, 2021, Disponível em: <https://devdocs.io/c/>. Acesso em 03/03/2021.

wikiHow, 2021, Disponível em: <https://pt.wikihow.com/Jogar-Senha>. Acesso em 12/02/2021.