ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital
Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Fundamentos Programação

Projeto nº 1

Sudoku

Realizado em: 09/01/2024

Francisco Crespo a2021138307

Índice

1. Introdução

O programa que será desenvolvido é um jogo de Sudoku, implementado em linguagem C, com base em menus de opções para proporcionar uma experiência interativa ao utilizador. Este jogo terá a capacidade de gerir até 100 problemas de Sudoku, seguindo as regras e referências fornecidas. A implementação do programa será organizada através do uso de funções em ficheiros separados, promovendo assim uma estrutura mais clara e compreensível.

2. Guia / Descrição do Jogo

O funcionamento do programa será guiado por menus, oferecendo opções ao utilizador para realizar diversas operações, como inserir ou remover números, abortar o jogo e retomar ao menu principal. Mensagens apropriadas serão apresentadas em resposta a cada operação, indicando se foi bem ou mal executada. Durante um jogo, apenas operações específicas serão permitidas, como inserção/remoção de números, garantindo assim a validade do Sudoku.

Para iniciar um novo jogo o utilizador poderá escolher entre duas opções, um problema já existente ou terá também a opção de um jogo aleatório.

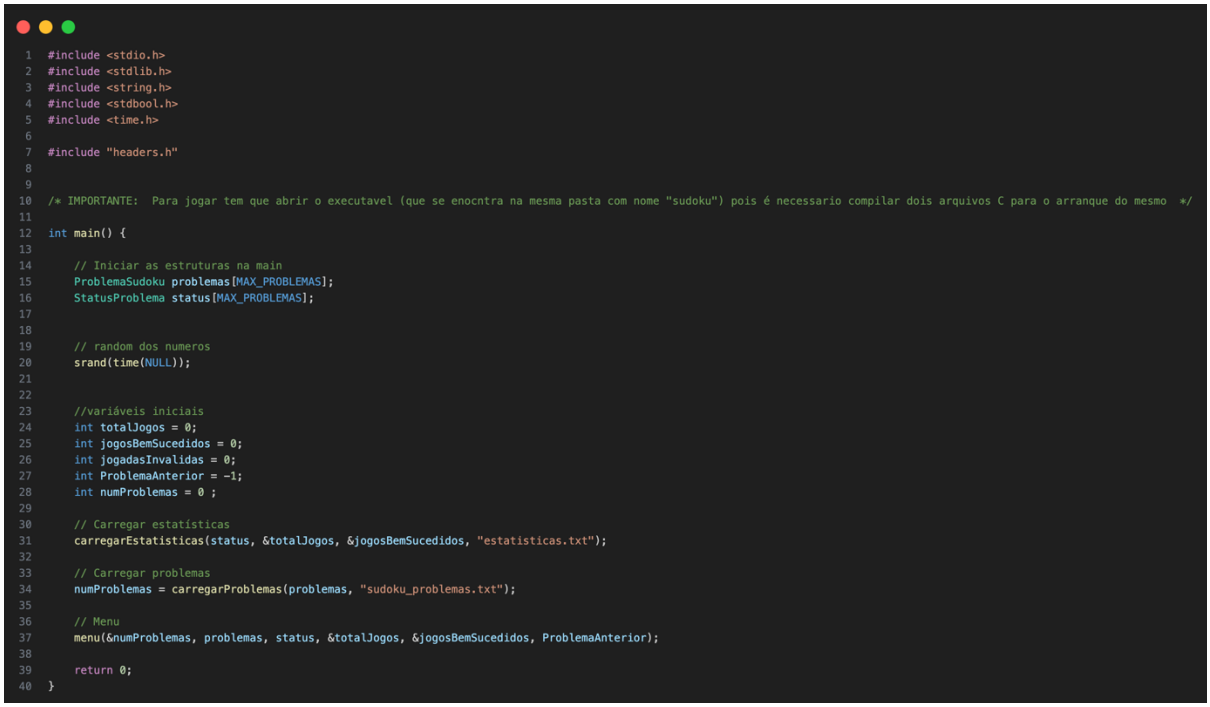
Para validar uma jogada, o programa seguirá as regras estabelecidas, garantindo que os números de 1 a 4 ocorram apenas uma vez em cada linha, coluna e quadrado 2x2.

A implementação será estruturada de forma a possibilitar a expansão do jogo até 100 problemas distintos, proporcionando variedade ao utilizador. Após a conclusão de um jogo, o utilizador terá a opção de iniciar um novo jogo ou retomar ao menu principal.

O programa também apresentará uma opção de estatísticas que irá apresentar ao utilizador as suas estatísticas, jogadas, jogadas invalidas e jogos concluídos com sucesso.

2.1. Main

O ficheiro principal, conhecido como "main", desempenha um papel crucial na execução do jogo. É neste arquivo que todas as funções necessárias para o funcionamento do jogo são chamadas. Além disso, é na "main" que as estruturas essenciais são inicializadas e as informações armazenadas em arquivos são carregadas. Este arquivo desempenha um papel central na organização e coordenação de todas as operações necessárias para o correto funcionamento do jogo, desde a configuração inicial até o carregamento dos dados fundamentais a partir de arquivos externos.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>

#include "headers.h"


/* IMPORTANTE:  Para jogar tem que abrir o executavel (que se enocntra na mesma pasta com nome "sudoku") pois é necessario compilar dois arquivos C para o arranque do mesmo  */

int main() {

    // Iniciar as estruturas na main
    ProblemaSudoku problemas[MAX_PROBLEMAS];
    StatusProblema status[MAX_PROBLEMAS];


    // random dos numeros
    srand(time(NULL));


    //variáveis iniciais
    int totalJogos = 0;
    int jogosBemSucedidos = 0;
    int jogadasInvalidas = 0;
    int ProblemaAnterior = -1;
    int numProblemas = 0 ;

    // Carregar estatísticas
    carregarEstatisticas(status, &totalJogos, &jogosBemSucedidos, "estatisticas.txt");

    // Carregar problemas
    numProblemas = carregarProblemas(problemas, "sudoku_problemas.txt");

    // Menu
    menu(&numProblemas, problemas, status, &totalJogos, &jogosBemSucedidos, ProblemaAnterior);

    return 0;
}
```

2.2. Funções

O ficheiro "funcoes.c" desempenha um papel crucial ao armazenar todas as funções essenciais para garantir o bom funcionamento do programa. Neste arquivo, são implementadas e executadas todas as operações necessárias para o correto desenvolvimento do programa. Aqui, encontram-se todas as funcionalidades cruciais que contribuem para a eficiência e desempenho adequado do sistema, proporcionando uma organização centralizada e facilitando a manutenção e compreensão do código-fonte.

Função Menu:

```c
#include "headers.h"

// menu

void menu(int *numProblemas, ProblemaSudoku problemas[MAX_PROBLEMAS], StatusProblema status[MAX_PROBLEMAS], int *totalJogos, int *jogosBemSucedidos, int ProblemaAnterior){

    int opcao;
    int escolha;

    do {
        printf("\n===== Sudoku =====\n");
        printf("1. Jogar Sudoku\n");
        printf("2. Adicionar Novo Problema\n");
        printf("3. Alterar Nome de um Problema\n");
        printf("4. Consultar Estatísticas\n");
        printf("5. Sair\n");
        printf("Escolha: ");
        scanf(" %d", &opcao);
        getchar();

        switch (opcao) {
            case 1:
                if(numProblemas > 0){
                    printf("Escolha o tipo de jogo:\n");
                    printf("1. Jogo Aleatorio\n");
                    printf("2. Escolher Jogo\n");
                    printf("3. Voltar\n");
                    printf("Escolha: ");
                    scanf("%d", &escolha);
                    getchar();
                    if (escolha == 1) {
                        int problemaSelecionado = escolherProblemaAleatorio(ProblemaAnterior, *numProblemas);
                        jogarSudoku(problemas, numProblemas, problemas[problemaSelecionado], status, totalJogos, jogosBemSucedidos, ProblemaAnterior);
                        ProblemaAnterior = problemaSelecionado;
                    } else if (escolha == 2) {
                        int problemaSelecionado = escolherJogo(*numProblemas, problemas);
                        jogarSudoku(problemas, numProblemas, problemas[problemaSelecionado], status, totalJogos, jogosBemSucedidos, ProblemaAnterior);
                        ProblemaAnterior = problemaSelecionado;
                    }else if(escolha == 3){
                        menu(numProblemas, problemas, status, totalJogos, jogosBemSucedidos, ProblemaAnterior);
                    }
                }else{
                        printf("Nao existem problemas de Sudoku disponiveis.\n");
                        adicionarProblema(problemas, numProblemas);
                    }
                break;
            case 2:
                // Adicionar novo problema
                    adicionarProblema(problemas, numProblemas);

                break;
            case 3:
                // Alterar o nome do problema
                alterarNomeProblema(problemas, *numProblemas);
                break;
            case 4:
                // Consultar estatisticas
                Estatisticas(status, *numProblemas, *totalJogos, *jogosBemSucedidos);
                break;
            case 5:
                // Sair e guardar estatisticas e problemas
                guardarEstatisticas(problemas, status, *numProblemas, *totalJogos, *jogosBemSucedidos, "estatisticas.txt");
                guardarProblemas(problemas, *numProblemas, "sudoku_problemas.txt");
                break;
            default:
                printf("Opção invalida. Tente novamente.\n");
        }

    } while (opcao != 5);

}
```

Função Escolher Jogo:

```c
// escolher tipo de jogo

int escolherJogo(int numProblemas, ProblemaSudoku problemas[MAX_PROBLEMAS]){
    int op;
    printf("Jogos:\n\n");
    for (int i = 0; i < numProblemas; i++)
    {
        printf("%d: %s",i+1,problemas[i].nome);
    }
    printf("\n\nEscolha o jogo que pretende jogar: ");
    scanf("%d", &op);
    getchar();

    return op-1;
}
```

Função Adicionar Problema:

```c
// adicionar problema sudoku
/* funcao que adiciona um sudoku , pedindo ao usuario para preencher o problema por linhas , tratamos o problema em matriz */

void adicionarProblema(ProblemaSudoku problemas[MAX_PROBLEMAS], int *numProblemas) {
    if (*numProblemas >= MAX_PROBLEMAS) {
        printf("Limite maximo de problemas atingido.\n");
        return;
    }

    printf("Digite o nome do novo problema: ");
    scanf("%s", problemas[*numProblemas].nome);
    getchar();
    printf("Digite os numeros do Sudoku (0 para posicao vazia):\n");
    for (int i = 0; i < TAMANHO; i++) {
        printf("Linha: %d\n", i + 1) ;
        for (int j = 0; j < TAMANHO; j++) {
            printf("Coluna %d: ", j + 1);
            scanf("%d", &problemas[*numProblemas].tabuleiro[i][j]);
        }
    }
    getchar();
    (*numProblemas)++;
    return;
}
```

Função Alterar Nome do Problema:

```c
// alterar nome
/* funcao que altera o nome a partir do nome existente, com comparacao de strings */

void alterarNomeProblema(ProblemaSudoku problemas[MAX_PROBLEMAS], int numProblemas) {
    char NomeAnterior[MAX_NOME_LENGTH];
    char NovoNome[MAX_NOME_LENGTH];

    printf("========== Lista de jogos ==========\n");
    for (int i = 0; i < numProblemas; i++)
    {
        printf("%d: %s\n",i+1,problemas[i].nome);
    }


    printf("Digite o nome do problema que deseja alterar: ");
    scanf("%s", NomeAnterior);
    getchar();
    int i;
    for (i = 0; i < numProblemas; i++) {
        if (strcmp(problemas[i].nome, NomeAnterior) == 0) {
            printf("Digite o novo nome: ");
            scanf("%s", NovoNome);
            getchar();
            // Verificar se o novo nome já existe
            int j;
            for (j = 0; j < numProblemas; j++) {
                if (j != i && strcmp(problemas[j].nome, NovoNome) == 0) {
                    printf("Ja existe um problema com esse nome.\n");
                    return;
                }
            }

            strcpy(problemas[i].nome, NovoNome);
            printf("Nome alterado com sucesso.\n");
            return;
        }
    }

    printf("Problema nao encontrado.\n");
}
```

Função para apresentar o problema:

```c
// mostrar Problema
/* funcao para mostrar o sudoko no interface do usuario */

void Problema(ProblemaSudoku problema) {
    printf("\n========== %s ==============\n", problema.nome);
    for (int i = 0; i < TAMANHO; i++) {
        for (int j = 0; j < TAMANHO; j++) {
            printf("- ", problema.tabuleiro[i][j]);
        }
        printf("\n");
    }
}
```

Função escolher problema aleatório:

```c
// escolher problema aleatorio
/* Onde o programa faz uma escolha aleatoria dos sudokus existentes */

int escolherProblemaAleatorio(int ProblemaAnterior, int numProblemas) {
    int problemaAleatorio;

    if (numProblemas <= 0) {
        printf("Nao existem problemas de Sudoku disponiveis.\n");
        exit(0);
    }

    do {
        problemaAleatorio = rand() % numProblemas;
    } while (problemaAleatorio == ProblemaAnterior);

    return problemaAleatorio;
}
```

Função jogar jogo:

```c
// jogar jogo
/*  onde se processa o jogo e as verificacoes */

void jogarSudoku(ProblemaSudoku problemas[MAX_PROBLEMAS], int *numProblemas ,ProblemaSudoku problema, StatusProblema *status, int *totalJogos, int *jogosBemSucedidos, int ProblemaAnterior) {
    int jogadasInvalidas = 0;

    while (true) {
        system("clear");

        /* mostrar o jogo na tela do usuario */
        Problema(problema);
        printf("\nSudoku nao esta completo.\n");

        int linha, coluna, num;
        printf("Digite a linha (1-%d), coluna (1-%d) e numero (1-%d) separados por espaço (0 0 0 para sair): ", TAMANHO, TAMANHO, TAMANHO);
        scanf("%d %d %d", &linha, &coluna, &num);

        /* sair do jogo */
        if (linha == 0 || coluna == 0 || num == 0) {
            break;
        }

        linha--;
        coluna--;

        if (linha < 0 || linha >= TAMANHO || coluna < 0 || coluna >= TAMANHO || num > TAMANHO || num < 0) {
            printf("Jogada invalida. Tente novamente.\n");
            continue;
        }

        if (jogadaValida(problema, linha, coluna, num)) {
            problema.tabuleiro[linha][coluna] = num;
            Problema(problema);
        } else {
            printf("Jogada invalida. Tente novamente.\n");
            jogadasInvalidas++;
            if (jogadasInvalidas >= 2) {
                printf("Fez duas jogadas invalidas. Jogo encerrado.\n");
                break;
            }
        }
        if (sudokuCompleto(problema)) {
                printf("Parabens! Completou o Sudoku.\n");
                printf("1. Sair\n2. Jogar novamente: ");
                int opcao;
                scanf("%d", &opcao);
                if (opcao == 2) {
                    int problemaSelecionado = escolherProblemaAleatorio(ProblemaAnterior, *numProblemas);
                    jogarSudoku(problemas, numProblemas, problemas[problemaSelecionado], status, totalJogos, jogosBemSucedidos, ProblemaAnterior);
                    ProblemaAnterior = problemaSelecionado;
                } else {
                    menu(numProblemas, problemas, status, totalJogos, jogosBemSucedidos, ProblemaAnterior);
                }
        }
    }


    (*totalJogos)++;
    if (jogadasInvalidas < 2) {
        (*jogosBemSucedidos)++;
    }

    for (int i = 0; i < MAX_PROBLEMAS; i++) {
        if (strcmp(status[i].nome, problema.nome) == 0) {
            status[i].jogadas++;
            status[i].jogadasInvalidas += jogadasInvalidas;
            strcpy(status[i].nome, problema.nome);
            break;
        }
    }
}
```

Função de verificação da jogada:

```c
// verificacao da jogada

int jogadaValida(ProblemaSudoku problema, int linha, int coluna, int num) {

    /* // Verificar se existe algum numero na posicao
    if (problema.tabuleiro[linha][coluna] != 0) {
        return 0;
    } */

    if(num != 0){
    // Verificar se o numero esta na mesma linha ou coluna
         for (int i = 0; i < TAMANHO; i++) {
            if (problema.tabuleiro[linha][i] == num || problema.tabuleiro[i][coluna] == num) {
                return 0;
                }
            }
        // Verificar se o numero esta na matriz 2x2
        int startlinha = (linha / 2) * 2;
        int startCol = (coluna / 2) * 2;
        for (int i = startlinha; i < startlinha + 2; i++) {
            for (int j = startCol; j < startCol + 2; j++) {
                if (problema.tabuleiro[i][j] == num) {
                    return 0;
                }
            }
        }
    }

    return 1;
}
```

Função que valida se o sudoku está completo:

Como o programa já faz a validação por jogada, esta validação final só necessita de validar se existe algum zero (0) no Problema.

```c
// verificacao se o sudoku esta completo
/* verifica se existem espaços por prencher, isto é, casas com zeros */

bool sudokuCompleto(ProblemaSudoku problema) {
    for (int i = 0; i < TAMANHO; i++) {
        for (int j = 0; j < TAMANHO; j++) {
            if (problema.tabuleiro[i][j] == 0) {
                return false;
            }
        }
    }
    return true;
}
```

Função para apresentar estatísticas:

```c
// estatisticas

void Estatisticas(StatusProblema status[MAX_PROBLEMAS], int numProblemas, int totalJogos, int jogosBemSucedidos) {
    printf("\n===== Estatisticas =====\n");

    for (int i = 0; i < numProblemas; i++) {
        printf("Problema: %s \nJogado: %d vez(es)\n Jogadas invalidas: %d\n", status[i].nome, status[i].jogadas, status[i].jogadasInvalidas);
    }

    printf("Numero total de jogos: %d\n", totalJogos);
    printf("Numero de jogos concluidos com sucesso: %d\n", jogosBemSucedidos);

    if (totalJogos > 0) {
        float successPercentage = ((float)jogosBemSucedidos / totalJogos) * 100;
        printf("Percentagem de sucesso: %.1f%%\n", successPercentage);
    } else {
        printf("Percentagem de sucesso: Nao existe\n");
    }
}
```

Funções para guardar em ficheiros:

Decidi guardar em dois ficheiros distintos, um para armazenar os problemas e outro para armazenar as estatísticas.

```c
// Guardar em ficherios


 /* Guardar Eststosticas */

void guardarEstatisticas(ProblemaSudoku problemas[MAX_PROBLEMAS], StatusProblema status[MAX_PROBLEMAS], int numProblemas, int totalJogos, int jogosBemSucedidos, char filename[]) {
    FILE *arquivo = fopen(filename, "w");
    if (!arquivo) {
        printf("Erro ao abrir o arquivo de estatisticas.\n");
        exit(0);
    }

    fprintf(arquivo, "Total de jogos: %d\nJogos Bem Sucedidos: %d\n", totalJogos, jogosBemSucedidos);

    for (int i = 0; i < numProblemas; i++) {
        fprintf(arquivo, "status: %s \njogadas:%d \njogadas invalidas: %d\n", status[i].nome, status[i].jogadas, status[i].jogadasInvalidas);
    }

    fclose(arquivo);
}

 /* Guardar Problemas */

void guardarProblemas(ProblemaSudoku problemas[MAX_PROBLEMAS], int numProblemas, char filename[]) {
    FILE *arquivo = fopen(filename, "w");
    if (!arquivo) {
        printf("Erro ao abrir o arquivo para guardar problemas.\n");
        exit(0);
    }

    for (int i = 0; i < numProblemas; i++) {
        fprintf(arquivo, "%s\n", problemas[i].nome);
        for (int j = 0; j < TAMANHO; j++) {
            for (int k = 0; k < TAMANHO; k++) {
                fprintf(arquivo, "%d ", problemas[i].tabuleiro[j][k]);
            }
            fprintf(arquivo, "\n");
        }
    }

    fclose(arquivo);
}
```

Funções para carregar ficheiros:

Como as informações do programa estão a ser armazenadas em dois ficheiros distintos, dessa forma o carregamento das informações para o programa também tem que ser realizado em duas funções distintas.

```c
// carregar ficheiros

/* Carregar estatisticas */

int carregarEstatisticas(StatusProblema status[MAX_PROBLEMAS], int *totalJogos, int *jogosBemSucedidos, char filename[]) {
    FILE *arquivo = fopen(filename, "r");
    if (!arquivo) {
        printf("Arquivo de estatisticas nao encontrado.\n");
        return 0;
    }

    fscanf(arquivo, "%d %d", totalJogos, jogosBemSucedidos);

    int i = 0;
    while (fscanf(arquivo, "%s %d", status[i].nome, &status[i].jogadas) == 2) {
        i++;
        if (i >= MAX_PROBLEMAS) {
            break;
        }
    }

    fclose(arquivo);
    return i;
}

/* Carregar Problemas */

int carregarProblemas(ProblemaSudoku problemas[MAX_PROBLEMAS], char filename[]) {
    FILE *arquivo = fopen(filename, "r");
    if (!arquivo) {
        printf("Arquivo de Problemas nao encontrado.\n");
        return 0;
    }

    int numProblemas = 0;
    while (fscanf(arquivo, "%s", problemas[numProblemas].nome) == 1) {
        for (int i = 0; i < TAMANHO; i++) {
            for (int j = 0; j < TAMANHO; j++) {
                fscanf(arquivo, "%d", &problemas[numProblemas].tabuleiro[i][j]);
            }
        }
        for (int i = 0; i < TAMANHO; i++) {
            for (int j = 0; j < TAMANHO; j++) {
                fscanf(arquivo, "%d", &problemas[numProblemas].solucao[i][j]);
            }
        }

        numProblemas++;
        if (numProblemas >= MAX_PROBLEMAS) {
            break;
        }
    }

    fclose(arquivo);
    return numProblemas;
}
```

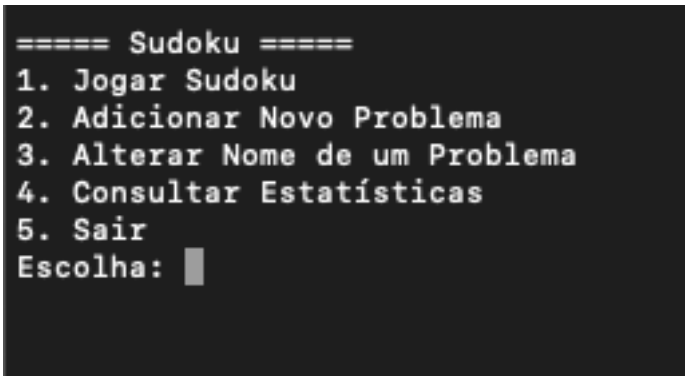
2.3. Headers

O ficheiro "headers.h" assume uma função fundamental na estruturação e organização de um programa. Este arquivo de cabeçalho/headers concentra-se na declaração de bibliotecas, definição de constantes, e inicialização de funções essenciais para o projeto. Ao centralizar estas informações, o "headers.h" promove uma abordagem modular e facilita a manutenção do código, permitindo uma visão rápida e abrangente das dependências e funcionalidades principais do programa.

```c
#ifndef HEADERS_H
#define HEADERS_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>

#define TAMANHO 4
#define MAX_PROBLEMAS 10
#define MAX_NOME_LENGTH 20
#define NUMEROS 10

// estruturas

typedef struct {
    char nome[MAX_NOME_LENGTH];
    int tabuleiro[TAMANHO][TAMANHO];
    int solucao[TAMANHO][TAMANHO];
} ProblemaSudoku;

typedef struct {
    char nome[MAX_NOME_LENGTH];
    int jogadas;
    int jogadasInvalidas;
} StatusProblema;

//  funções

void menu(int *numProblemas, ProblemaSudoku problemas[MAX_PROBLEMAS], StatusProblema status[MAX_PROBLEMAS], int *totalJogos, int *jogosBemSucedidos, int ProblemaAnterior);

void adicionarProblema(ProblemaSudoku problemas[MAX_PROBLEMAS], int *numProblemas);

void alterarNomeProblema(ProblemaSudoku problemas[MAX_PROBLEMAS], int numProblemas);

int escolherJogo(int numProblemas, ProblemaSudoku problemas[MAX_PROBLEMAS]);

int escolherProblemaAleatorio(int ProblemaAnterior, int numProblemas);

void Problema(ProblemaSudoku problema);

void jogarSudoku(ProblemaSudoku problemas[MAX_PROBLEMAS], int *numProblemas, ProblemaSudoku problema, StatusProblema *status, int *totalJogos, int *jogosBemSucedidos, int ProblemaAnterior);

int jogadaValida(ProblemaSudoku problema, int linha, int coluna, int num);

bool sudokuCompleto(ProblemaSudoku problema);

int carregarProblemas(ProblemaSudoku problemas[MAX_PROBLEMAS], char filename[]);

int carregarEstatisticas(StatusProblema status[MAX_PROBLEMAS], int *totalJogos, int *jogosBemSucedidos, char filename[]);

void guardarEstatisticas(ProblemaSudoku problemas[MAX_PROBLEMAS], StatusProblema status[MAX_PROBLEMAS], int numProblemas, int totalJogos, int jogosBemSucedidos, char filename[]);

void guardarProblemas(ProblemaSudoku problemas[MAX_PROBLEMAS], int numProblemas, char filename[]);

void Estatisticas(StatusProblema status[MAX_PROBLEMAS], int numProblemas, int totalJogos, int jogosBemSucedidos);

#endif
```

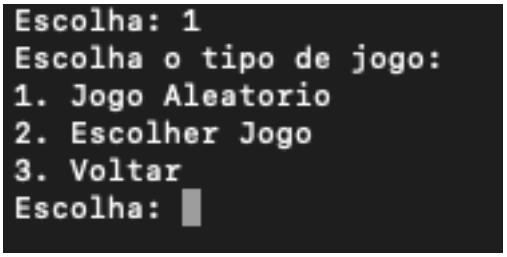
3. Modo Funcionamento

Menu Principal

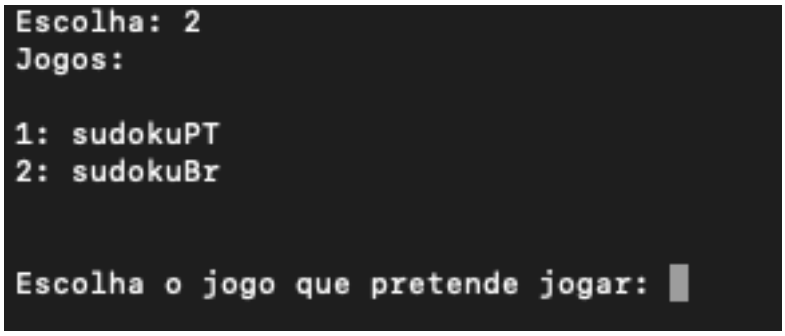


Opção 1. Jogar Sudoku

O Utilizar vai puder escolher um problema para jogar ou ainda tem a opção de jogar um jogo aleatório.



Caso pretenda jogar um jogo em específico, é lhe apresentado uma lista com todos os problemas e ele pode escolher o jogo utilizando o número respetivo.



Após a escolha do problema ou modo aleatório, o utilizador vai entrar no jogo, e é lhe apresentada a seguinte tela, caso o utilizador deseje sair tem que inserir os 4 campos a 0.

```
========== sudokuPT =============
 0  1  0  0
 0  0  2  0
 3  0  0  0
 0  0  0  4

Sudoku nao esta completo.
Digite a linha (1-4), coluna (1-4) e numero (1-4) separados por espaço (0 0 0 para sair):
```

Caso o utilizador faça mais que duas faltas, o jogo termina e o utilizador não conseguiu terminar o problema.

```
========== sudokuPT =============
 0  1  0  0
 0  0  2  0
 3  0  0  0
 0  0  0  4

Sudoku nao esta completo.
Digite a linha (1-4), coluna (1-4) e numero (1-4) separados por espaço (0 0 0 para sair): 1 2 1
Jogada invalida. Tente novamente.
Fez duas jogadas invalidas. Jogo encerrado.

===== Sudoku =====
1. Jogar Sudoku
2. Adicionar Novo Problema
3. Alterar Nome de um Problema
4. Consultar Estatísticas
5. Sair
Escolha:
```

Caso o utilizador tenha conseguido terminar o sudoku, respeitando as regras e passando por todas as validações é lhe apresentado uma mensagem de sucesso, e se deseja continuar a jogar ou sair para o menu principal. Caso decida jogar novamente, será lhe apresentado um novo problema, mas desta vez aleatoriamente.

```
========== sudokuBr =============
 1  3  4  2
 4  2  1  3
 2  4  3  1
 3  1  0  4

Sudoku nao esta completo.
Digite a linha (1-4), coluna (1-4) e numero (1-4) separados por espaço (0 0 0 para sair): 4 3 2

========== sudokuBr =============
 1  3  4  2
 4  2  1  3
 2  4  3  1
 3  1  2  4
Parabens! Completou o Sudoku.
Digite 1 para Sair ou 2 para jogar novamente:
```

Opção 2. Adicionar Problema

Este menu servirá para adicionar um problema ao programa, adicionando o sudoku por linhas.

```
Escolha: 2
Digite o nome do novo problema: sudokuTeste
Digite os numeros do Sudoku (0 para posicao vazia):
Linha: 1
Coluna 1: 0
Coluna 2: 2
Coluna 3: 0
Coluna 4: 0
Linha: 2
Coluna 1: 1
Coluna 2: 0
Coluna 3: 0
Coluna 4: 0
Linha: 3
Coluna 1: 0
Coluna 2: 0
Coluna 3: 3
Coluna 4: 0
Linha: 4
Coluna 1: 4
Coluna 2: 0
Coluna 3: 0
Coluna 4: 0
```

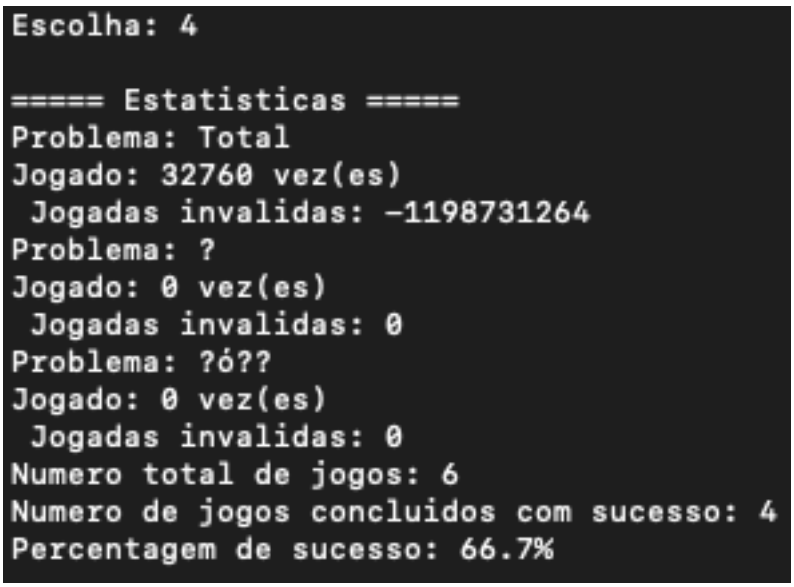
Opção 3. Alterar Nome do Problema

Nesta opção é permitido ao utilizador alterar o nome de um problema já existente, introduzindo o nome do problema que deseja alterar e o novo nome que pretende inserir.

```
Escolha: 3
========== Lista de jogos ===========
1: sudokuPT
2: sudokuBr
3: sudokuTeste
Digite o nome do problema que deseja alterar: sudokuTeste
Digite o novo nome: sudokuTeste11
Nome alterado com sucesso.
```

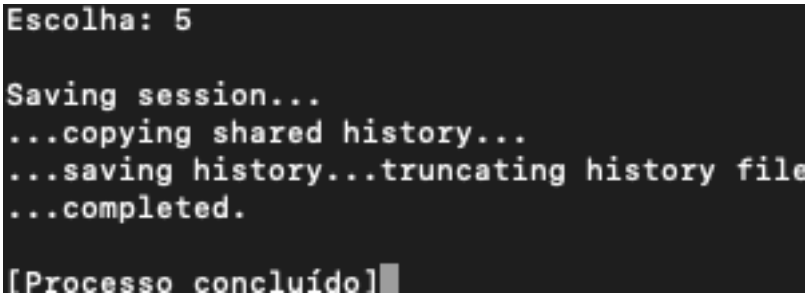
Opção 4. Estatísticas

Esta opção retorna ao utilizador as suas estatísticas, por problema e totais.



Opção 5. Sair

Opção para sair do programa e guardar todas as informações.



4. Conclusões

Em síntese, o programa em C desenvolvido para jogar Sudoku 4x4 oferece funcionalidades abrangentes, desde o armazenamento e manipulação de problemas até a validação interativa das jogadas. Com menus intuitivos, validação rigorosa e mensagens informativas, proporciona uma experiência eficaz e completa. A capacidade de gerenciar até 100 problemas e as consultas estatísticas adicionam versatilidade ao programa, tornando-o uma solução interativa e funcional para amantes do Sudoku.

Referências

1. WikiBooks:

   https://pt.wikibooks.org/wiki/Matem%C3%A1tica_divertida/Mini-sudoku

2. Sudokuonline:

   https://www.sudokuonline.io/pt/criancas/numeros-4-4