



**UFSJ – UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI**

**ALGORITIMO E ESTRUTURA DE DADOS I**

**SUDOKU**

**PROF.: DR. VINICIUS H. S. DURELLI**

**ANDRÉ SANTOS**

**MATHEUS NASCIMENTO**

**MATHEUS NATANAEL**

**São João del-Rei, 09 de dezembro de 2017**

## SUMÁRIO

<b>INTRODUÇÃO .....</b>	<b>3</b>
<b>ORIGEM DO JOGO .....</b>	<b>4</b>
<b>INSTRUÇÕES.....</b>	<b>5</b>
MENU .....	5
CARREGUE SEU JOGO .....	6
JOGAR UM JOGO ALEATORIO.....	6
RECORDE .....	6
COMO JOGAR.....	7
INSTRUÇÕES .....	7
SAIR.....	7
<b>COMO JOGAR .....</b>	<b>8</b>
INSTRUÇÕES BASICAS .....	8
AS POSSIBILIDADES.....	8
TIRANDO DA RETA .....	8
<b>INDEX .....</b>	<b>9</b>
<b>MODO 1 (CARREGUE SEU JOGO).....</b>	<b>10</b>
<b>MODO 2 (JOGAR UM JOGO ALEATORIO) .....</b>	<b>13</b>
<b>PEGA_RECORD.....</b>	<b>14</b>
<b>RECORD .....</b>	<b>18</b>
<b>CONSIDERAÇÕES FINAIS .....</b>	<b>19</b>
<b>REFERENCIA .....</b>	<b>20</b>

## INTRODUÇÃO

Foi proposto como trabalho pratico da disciplina de Algoritmo e estrutura de dados o desenvolvimento de um Sudoku.

Através deste trabalho pudemos testar e aprimorar os conhecimentos adquiridos durante a disciplina.

Durante a implementação do código fonte do jogo surgiram várias dúvidas que proporcionaram novos aprendizados.

## ORIGEM DO JOGO

Em bom japonês, o nome Sudoku é uma simplificação da frase “suji wa dokushin ni kagiru”, que significa “os números têm que ser únicos” e se refere a um passatempo numérico de instruções bem simples que exige lógica e raciocínio para a resolução.

Apesar do nome, o Sudoku (lê-se sudôku) não foi criado no Japão. A invenção é creditada ao matemático suíço Leonhard Euler. No século 18, ele criou o que chamou de “quadrados latinos”, um jogo em que os algarismos devem aparecer apenas uma vez em cada linha e em cada coluna. O formato com 9 linhas e 9 colunas se tornou popular quando começou a ser publicado nos EUA, na década de 1970.

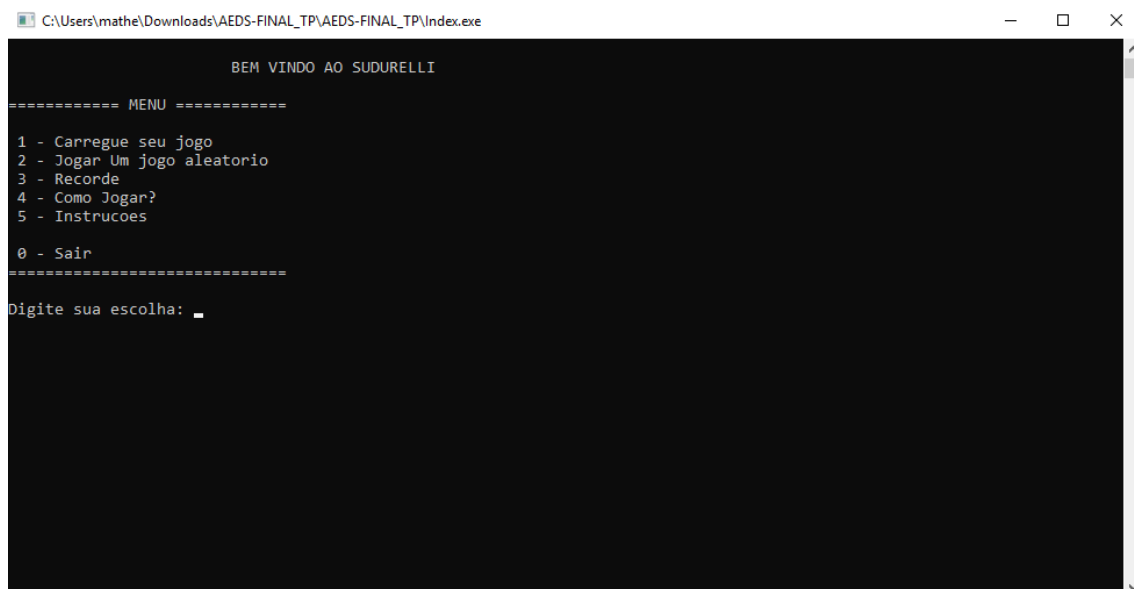
Foi lá que, em 1984, o japonês Maki Kaji conheceu a brincadeira. Ao voltar para sua terra natal, Kaji aprimorou o jogo (deu ordem aos números que já aparecem no quadrado e criou diferentes graus de dificuldade), o batizou e o transformou em uma febre entre seus conterrâneos (hoje o Japão tem mais de 600 mil revistas especializadas em sudoku).

No Ocidente, o jogo só virou mania em 2005. O primeiro passo para isso foi dado em 1997, quando o neozelandês Wayne Gould visitou o Japão, conheceu o sudoku e desenvolveu um programa de computador para o jogo, lançado em 2004.

## INSTRUÇÕES

Obs.: Deve-se compilar os arquivos Index.c, Modo1.c, Modo2.c antes de executar o jogo.

- **Menu:**



```
C:\Users\mathe\Downloads\AEDS-FINAL_TP\AEDS-FINAL_TP\Index.exe
BEM VINDO AO SUDURELLI

===== MENU =====
1 - Carregue seu jogo
2 - Jogar Um jogo aleatorio
3 - Recorde
4 - Como Jogar?
5 - Instrucoes

0 - Sair
=====
Digite sua escolha: _
```

Logo quando o jogo é executado aparece o menu com as seguintes opções:

1. Carregue seu jogo
2. Jogar um jogo aleatório
3. Recorde
4. Como jogar
5. Instruções
0. Sair

- **Carregue seu jogo:**

Nessa opção o programa carrega um jogo específico.

O usuário deve colocar um arquivo .txt com o nome “sudoku1.txt” antes de escolher essa opção, pois ela carrega esse arquivo para que o mesmo possa ser disponibilizado pelo usuário para ser jogado.

Após o carregamento do arquivo o usuário escolherá o grau de dificuldade em que se deseja jogar.

Obs.: o arquivo deve estar composto de todos os 81 números preenchidos de forma correta para que esse modo funcione.

- **Jogar um jogo aleatório:**

Nessa opção o programa carrega um jogo aleatório pré-definido pelo programa, assim como no modo anterior o jogo é carregado de um arquivo .txt, sendo esses arquivos já escolhidos pela equipe de desenvolvimento.

Após escolher esse modo o usuário escolherá o grau de dificuldade em que se deseja jogar.

- **Recorde:**

Nessa opção será apresentado para o usuário os três melhores jogadores de acordo com o tempo.

Obs. São apresentados os recordes sem distinção de dificuldade.

- **Como jogar:**

Nessa opção são apresentadas as regras básicas do jogo tais quais são apresentadas na pag. 7 desse relatório.

- **Instruções:**

Nessa opção são apresentadas as funcionalidades do programa.

- **Sair:**

Nessa opção o usuário sai do jogo

## COMO JOGAR

### 1. Instruções básicas

O jogador precisa distribuir, num quadrado de 81 casas, os números 1, 2, 3, 4, 5, 6, 7, 8 e 9. Não é preciso fazer nenhuma conta, basta espalhar os algarismos sem repeti-los na horizontal, na vertical e nos quadrados menores (de 9 células). Só existe uma solução certa para cada casa (na abreviação, Su quer dizer número e Doku, único).

### 2. As possibilidades

O primeiro passo é analisar cada linha, coluna e célula e encontrar os números que poderiam ser colocados ali, ou seja, aqueles que ainda não existem em nenhuma dessas 3 posições. Comece sempre pelos grupos que têm mais números-pista já dispostos. O ideal é anotar todas as possibilidades a lápis, para poder ir apagando depois.

### 3. Tirando da reta

Nas casas em que só há uma possibilidade, você já tem o resultado. Escreva o número e exclua-o das outras casas que estejam na mesma linha, coluna ou célula. Repita o processo várias vezes, até preencher todo o quadrado.



## INDEX

No Index ocorre a primeira chamada do programa, nele é apresentado o Menu onde há as funções acima mencionado.

Usamos a biblioteca *colors.h* para adicionar cores ao programa.

Utilizamos do *while* para a criação do laço de repetição do Menu e o *Switch case* para a definição de cada escolha feita.

Nos cases 1, 2 e 3 onde há a necessidade da abertura de outro arquivo .c existe o *system("./")* onde ocorre a abertura do outro arquivo.

Há também um *printf("\e[H\e[2J")* que é utilizado para a limpeza da tela em Linux.

## MODO 1 (CARREGUE SEU JOGO)

No modo 1 foram declarados 3 tabuleiros com funções diferentes.

O primeiro tabuleiro recebe, do arquivo inserido pelo usuário, a matriz 9x9 utilizada no game, o tabuleiro1 recebe o mesmo arquivo, e será utilizada para a conferência da tabela a cada jogada e ao final do jogo. O tabuleiro3 recebera o tabuleiro original após ter sido sorteada as posições que serão ocultadas a fim de ser utilizado como parâmetro para o bloqueio dos números iniciais do jogo.

A primeira função chamada nesse modo é a função “*inicializaTabuleiro*” que passa todos os 3 tabuleiros e o nível de dificuldade para que sejam realizados a inicialização, sorteio e preenchimento das tabelas.

Nessa função é necessário a declaração de um ponteiro (*arq*), que auxiliará na execução da função.

Através do comando *FILE \*arq = fopen ("sudoku1.txt", "r")* abrimos o arquivo .txt onde esta localizado a matriz 9x9 utilizada no jogo.

A partir do momento quando ocorre o carregamento do jogo o tempo do cronometro é acionado, começando a contar o tempo de jogo do usuário, essa ação é executada pelo comando *inicio=(int)time(NULL)*.

Logo após esse comando ser executado aparece o primeiro *for* onde ocorre o preenchimento do primeiro tabuleiro.

O segundo *for* é utilizado para o preenchimento do tabuleiro1 como a cima mencionado.

Na sequencia aparece um conjunto de *if's* e *else's* onde ocorre o sorteio das posições a serem ocultadas da matriz que será apresentada ao usuário.



função é novamente chamada e solicitado ao usuário que entre com os valores novamente.

```
void pegaValores(int valor[3]){ //funcao que pega valores e confere se sao validos

    printf("Digite a linha: ");
    scanf("%d",&valor[0]);
    printf("\nDigite a coluna: ");
    scanf("%d",&valor[1]);
    printf("\nValor: ");
    scanf("%d",&valor[2]);
    if (valor[0]<0 || valor[0]>tam){
        printf("\nLinha invalida, por favor digite numeros entre 1 e 9\n");
        pegaValores(valor);
    }
    if (valor[1]<0 || valor[1]>tam){
        printf("\nColuna invalida, por favor digite numeros entre 1 e 9\n");
        pegaValores(valor);
    }
    valor[0]--;
    valor[1]--;
    if (valor[2]<-1 || valor[2]>tam){
        printf("\nValor invalido, por favor digite numeros entre 1 e 9\n");
        pegaValores(valor);
    }
}
```

Na função seguinte ocorre a verificação se a posição digitada pelo usuário é uma posição valida, ou seja, se é uma posição onde não havia previamente um número disposto.

Essa verificação ocorre através da conferencia do tabuleiro2 (onde havia sido colocado as posições ocultas). Se a posição estivesse “vazia” o número digitado é atribuído à posição, caso contrario a modificação da tabela original não ocorre.

A seguinte função utiliza-se três parâmetros, o tabuleiro, tabuleiro1 e o vetor valor. Nessa função ocorre a comparação dos valores digitados com o tabuleiro de resposta, se o usuário digitou um numero corretamente a função printa o tabuleiro em verde, se o numero estiver errado o tabuleiro é printado em vermelho. Para tal feito utilizamos novamente a biblioteca *colors.h* e o comando *foreground (COR DESEJADA)*.

Ao final do jogo a função `comparacaofinal` entra em campo para conferir se o jogo está todo correto caso essa verificação esteja correta o tempo de jogo é parado e a função seguinte é chamada.

Através da função `recorde` o tempo de jogo é salvo em um arquivo `.txt`, e o arquivo de comparação de tempo é chamado.

## MODO 2 (JOGAR UM JOGO ALEATORIO)

O modo 2 tem as mesmas funcionalidades presentes no modo 1, todas as funções são idênticas, a única diferença é que nesse modo o jogo escolhe aleatoriamente qual jogo será jogado. Para esse sorteio foi novamente empregado o comando `rand`.

```
void inicializaTabuleiro(int tabuleiro[][tam],
    int tabuleiro1[][tam], int tabuleiro2[][tam], int nivel){ //função que inicializa o jogo do usuário

int linha, coluna,num, i, j,aleat=0;
FILE *arq;
srand(time(NULL));

    aleat=rand()%100;

    if(aleat%2==0){
        arq=fopen("sudoku1.txt", "r");
        if((arq = fopen("sudoku1.txt", "r"))!=0){goto abriu;}
    }
    if(aleat%3==0){
        arq=fopen("sudoku2.txt", "r");
        if((arq = fopen("sudoku2.txt", "r"))!=0){goto abriu;}
    }
    if(aleat%5==0){
        arq=fopen("sudoku3.txt", "r");
        if((arq = fopen("sudoku3.txt", "r"))!=0){goto abriu;}
    }
    else{
        arq=fopen("sudoku4.txt", "r");
        if((arq = fopen("sudoku4.txt", "r"))!=0){goto abriu;}
    }

    printf("Problemas na abertura do arquivo\n");
    printf("Arquivo nao encontrado\n");
    exit(1);
```

## **PEGA\_RECORD**

Esse arquivo é o responsável pela comparação e adição de novos recordes.

Como mencionado no Modo 1 o tempo de jogo é salvo em um arquivo .txt, através desse arquivo podemos comparar se o tempo de jogo do usuário foi menor q o tempo de jogo de qualquer outro jogador já presente nos recordes do jogo.

Foram declarados nesse arquivo .c duas estruturas a time para ser armazenado o tempo de jogo do usuário, e a record onde são armazenados os nomes e tempos de jogo presentes no arquivo *record.txt*.

Ah nesse arquivo .c quatro funções carregar\_record, comparar\_record, pegar\_record\_novo, gravar\_record.

Na função carrega\_record são carregados os arquivos .txt, o recorde.txt onde está presente o tempo do usuário e o record.txt onde estão os recordes salvos.

```
void carrega_record(struct record record[tam], struct time time[1]){
    int i;

    FILE *arq = fopen("record.txt", "r+");
    FILE *arq1 = fopen("recorde.txt", "r+");

    if(arq==NULL || arq1==NULL){

        printf("Problemas na abertura do arquivo\n");
        printf("Arquivo nao encontrado\n");

    } else{

        fscanf(arq1, "%d", &time[1].tempo);

        for(i=0; i<tam; i++){
            fscanf(arq, "%s", &record[i].nome);
            fscanf(arq, "%d", &record[i].tempo);
        }

        fclose(arq);
        fclose(arq1);
    }
}
```

Para isso declaramos dois ponteiros para auxiliar na manipulação dos arquivos .txt, *arq* e *arq1*.

Utilizamos *fscanf* para preencher as estruturas com as informações presentes nos arquivos.

Após a utilização dos arquivos .txt foi utilizado *fclose(ponteiro)* para o fechamento dos mesmos.

Na sequencia é chamada a função *compara\_record*, onde é comparado se o tempo atingido pelo usuário é menor que o tempo presente em um dos records salvos.

Se o tempo do usuário for menor a função *pegar\_record\_novo* é chamada e recebe como parâmetro as *struct's time* e *record* e o *índice* do recorde de maior tempo.

```
void gravar_record(struct record record[tam]){  
  
void pegar_record_novo(struct time time[1], struct record record[tam], int i){  
  
    setbuf(stdin, NULL);  
    printf("\nPARABENS NOVO RECORD, DIGITE O SEU PRIMEIRO NOME");  
    printf("\nNome: ");  
    fgets(record[i].nome, 100, stdin);  
    record[i].tempo = time[1].tempo;  
  
    gravar_record(record);  
  
}
```

Na função `pegar_record_novo` é solicitado ao usuário que digite seu primeiro nome, para que não ocorra erros limpamos o buffer do teclado com o comando `setbuf (stdin, NULL)`. Após o usuário digitar seu nome o mesmo é adicionado à struct na posição indicada pelo índice, assim como o seu tempo. Na sequência a função `gravar_record` é chamada.

A função `gravar_record` é a responsável por atualizar o arquivo `record.txt` com os novos registros



```
void gravar_record(struct record record[tam]){  
  
    int i,j;  
    FILE *arq = fopen("record.txt", "w");  
    if(arq==NULL){  
        printf("Problemas na abertura do arquivo\n");  
        printf("Arquivo nao encontrado\n");  
  
    } else{  
  
        for(i=0; i<tam; i++){  
            for(j=0; j<strlen(record[i].nome); j++){  
                fputc(record[i].nome[j], arq);  
            }  
            fprintf(arq, " %d\n\n", record[i].tempo);  
        }  
  
        printf("\nRecord Atualizado");  
        sleep(2);  
    }  
  
    fclose(arq);  
    system("./Record");  
}
```

Utilizamos dois *for's* para fazer o laço necessários para realizar a escrita no arquivo record.txt.

Dentro do segundo laço for foi utilizado o comando *fputc(record[i].nome[j], arq)* para escrever os caracteres dos nomes dos jogadores e o comando *fprintf (arq, " %d\n\n", record[i].tempo)* para escrever o tempo de cada um. Mais uma vez o *fclose* é utilizado para fechar o arquivo .txt.

Em sequencia é chamado o arquivo Record.c através do comando *system("./Record")*.

## RECORD

Esse arquivo é responsável pela apresentação dos recordes, nele estão presentes duas funções uma para o carregamento dos recordes presentes no arquivo record.txt e uma para printar tais recordes na tela.

A função `carrega_recorde` é a responsável pelo carregamento do arquivo record.txt, ela abre o arquivo e o adiciona em uma estrutura de nome `record`. Após esse carregamento é chamada a função `ler_record`, responsável por printar na tela para o usuário os recordes do jogo.

```
void carrega_record( struct record record[tam]){
    int i;
    FILE *arq = fopen("record.txt", "r");
    for(i=0; i<tam; i++){
        fscanf(arq, "%s", &record[i].nome);
        fscanf(arq, "%d", &record[i].tempo);
    }
}

void ler_record(struct record record[tam]){
    int i, b;
    printf("\n===== RECORD =====");
    for(i=0; i<tam; i++){
        printf("\n\n Nome: %s\n", record[i].nome);
        printf(" Tempo de jogo: %d Segundos\n", record[i].tempo);
        printf("-----\n");
    }
    printf("\n===== \n\n");
    printf("Deseja voltar ao menu? (0 - Nao / 1 - Sim): ");
    scanf("%d", &b);
    if(b==1){
        printf("\e[H\e[2J");
        system("./Index");
    }
    else{
        exit;
    }
}
```

## **CONSIDERAÇÕES FINAIS**

O trabalho proposto só foi concluído devido a dedicação de cada membro do grupo, codificando e auxiliando os demais com seu conhecimento.

Devido à complexidade e dificuldade encontrada durante a produção do código foram necessárias algumas pesquisas.

O comando rand foi encontrado em um fórum (link está nas referências), onde havia um código pronto, mas não atendia ao problema encontrado, sendo assim um novo código foi escrito de acordo com os conhecimentos adquiridos na video aula.

A biblioteca colors.h é de autoria de Leônidas S. Barbosa, a mesma foi encontrada em um fórum na internet (link está nas referências).

## REFERENCIA

- **História do Sudoku:**

*<https://super.abril.com.br/historia/o-que-e-sudoku/>*

- **Video aula Rand:**

*<http://linguagemc.com.br/valores-aleatorios-em-c-com-a-funcao-rand/>*

- **Lendo uma Matriz quadrada por um Arquivo .TXT:**

*[https://www.youtube.com/watch?v=tNH\\_ecdFTpc](https://www.youtube.com/watch?v=tNH_ecdFTpc)*

*<http://codigosfontes-ccplus-plus.blogspot.com.br/2014/03/lendo-uma-matriz-quadrada-por-um.html>*

- **Biblioteca Colors.h:**

*<https://www.youtube.com/watch?v=kIPwvR3FihM>*

*<https://github.com/kirotawa/Scripts/blob/master/C/colors/src/colors.h>*