```c
//Código produzido por:
//Felipe Ferreira Campos
//Fernando dos Santos Figueredo
//Raphael Prata
//Paulo Henrique Abreu Neiva
//Anita Pereira
//Marcus Dornas

//Trabalho de Programação Imperativa
//Sudoku v1.10

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define SO 1 //SO = 1 quer dizer que é Windows, SO = 0 quer dizer que é Linux, tem qua alterar isso pq se não a funcao de limpar da erro

//Prototipo das funções que serão usadas na correção
void limpaTela();
void imprimiGradeJogo();
void imprimiGradeCerta();
void imprimiRegraJogo();
int validaLinha(int, int, int);

// Preenchimento inicial do Sudoku
int grade[9][9] =
  {{ 0, 6, 0, 1, 0, 4, 0, 5,  0},
   { 0, 0, 8, 3, 0, 5, 6, 0, 0},
   { 2, 0, 0, 0, 0, 0, 0, 0, 1},
   { 8, 0, 0, 4, 0, 7, 0, 0, 6},
   { 0, 0, 6, 0, 0, 0, 3, 0, 0},
   { 7, 0, 0, 9, 0, 1, 0, 0, 4},
   { 5, 0, 0, 0, 0, 0, 0, 0, 2},
   { 0, 0, 7, 2, 0, 6, 9, 0, 0},
   { 0, 4, 0, 5, 0, 8, 0, 7, 0}};

int main()
{
    int sair, linha, coluna, valor;

    /*O do fará com que o programa rode pelo menos uma vez
    perguntado no final se o usuario deseja continuar jogando ou não*/
    do
    {
        int num;
        /*entrada de dados no jogo*/
        imprimiGradeJogo();
        printf("\n Digite as coordenadas do campo que deseja preencher");
        printf("\n Digite -1 na Linha para Sair");
        //printf("\n Digite -2 na Linha visualizar o jogo pronto."); É um cheat do jogo.
        printf("\n Digite -999 na linha para ler as regras do jogo");
        printf("\n\n Linha: ");
        scanf("%d", &linha);

        if(linha == -1){
            sair = 1;
        }else if(linha == -2){
            imprimiGradeCerta();
            printf("\n");
            system("pause");
        }else if(linha == -999){
            imprimiRegraJogo();
            printf("\n");
            system("pause");
        }
```

```
 65          else{
 66              printf(" Coluna: ");
 67              scanf("%d", &coluna);
 68              printf("\n Digite o valor a ser inserido na linha %d coluna %d:
", linha, coluna);
 69              scanf("%d", &valor);
 70
 71              /*verifica a escolha do usuario*/
 72              if((valor < 1)||(valor > 9)){
 73                  printf("\n*** Valor invalido, digite um valor entre 1 e 9
***");
 74                  printf("\n");
 75                  system("pause");
 76              }
 77              else
 78              {
 79                  /*verifica se o número já foi digitado na linha e coluna*/
 80                  if(validaLinha((linha-1), 0, valor) || validaColuna((coluna-
1), 0, valor)){
 81                      printf("\nValor ja foi digitado");
 82                      printf("\n");
 83                      system("pause");
 84                  }else{
 85                      grade[linha-1][coluna-1] = valor;
 86                  }
 87              }
 88
 89          }
 90
 91          printf("\n\n");
 92      } while(sair!=1);
 93  }
 94  /*usado para se jogar novamente*/
 95  void limpaTela(){
 96      if(SO)
 97          system("cls"); // Se SO = 1 Windows
 98      else
 99          system("clear"); //Se SO = 0 Linux
100  }
101
102  void imprimiGradeJogo(){
103      int i,j;
104      limpaTela();
105      system("color 4e"); //desativar essa linha caso for rodar no linux
106      FILE *arq;
107      char Linha[100];
108      char *result;
109      int l;
110      // Abre um arquivo TEXTO para LEITURA
111      arq = fopen("logo.txt", "r");
112      if (arq == NULL)  // Se houve erro na abertura
113      {
114          printf("Problemas na abertura do arquivo\n");
115          return;
116      }
117      l = 1;
118      while (!feof(arq))
119    {
120      // Lê uma linha (inclusive com o '\n')
121      result = fgets(Linha, 100, arq);  // o 'fgets' lê até 99 caracteres ou
até o '\n'
122      if (result)  // Se foi possível ler
123      printf(Linha);
124      l++;
125    }
126      fclose(arq);
```

```
127
128       printf("\n\nCreditos: Fernando, Raphael, Felipe, Anita, Paulo,
Marcus\n\n\n");
129       printf("    1  2  3   4  5  6   7  8  9  \n");//Desenho inicial do
sudoku
130       printf("  +---------+---------+---------+\n");
131       for (i = 0; i < 9; i++)
132       {
133           printf("%d |", i+1); //Barra lateral esquerda
134
135           for (j = 0; j < 9; j++)
136           {
137               if (grade[i][j] != 0) //Mostra os campos existentes
138                   printf(" %d ", grade[i][j]);
139               else
140                   printf("   ");
141
142               if (j % 3 == 2)// Após 3 alementos na coluna printa uma barra
143                   printf("|");
144           }
145
146           if (i % 3 == 2)// Após 3 elementos na linha print a divisoria
147           printf("\n  +---------+---------+---------+");
148           printf("\n");
149       }
150   }
151
152   void imprimiGradeCerta(){
153
154       limpaTela();
155       system("color 1f");
156       FILE *arq;
157       char Linha[100];
158       char *result;
159       int i;
160       // Abre um arquivo TEXTO para LEITURA
161       arq = fopen("sudoku.txt", "r");
162       if (arq == NULL)  // Se houve erro na abertura
163       {
164           printf("Problemas na abertura do arquivo\n");
165           return;
166       }
167       i = 1;
168       while (!feof(arq))
169     {
170       // Lê uma linha (inclusive com o '\n')
171       result = fgets(Linha, 100, arq);  // o 'fgets' lê até 99 caracteres ou
até o '\n'
172       if (result)  // Se foi possível ler
173       printf(Linha);
174       i++;
175     }
176       fclose(arq);
177     }
178
179   void imprimiRegraJogo(){
180
181       limpaTela();
182       system("color 2f");
183       FILE *arq;
184       char Linha[100];
185       char *result;
186       int i;
187       // Abre um arquivo TEXTO para LEITURA
188       arq = fopen("regra.txt", "r");
189       if (arq == NULL)  // Se houve erro na abertura
```

```
190       {
191           printf("Problemas na abertura do arquivo\n");
192           return;
193       }
194       i = 1;
195       while (!feof(arq))
196     {
197       // Lê uma linha (inclusive com o '\n')
198       result = fgets(Linha, 100, arq);  // o 'fgets' lê até 99 caracteres ou
até o '\n'
199       if (result)  // Se foi possível ler
200       printf(Linha);
201       i++;
202     }
203       fclose(arq);
204     }
205
206   int validaLinha(int linhaDigita, int colunaTestar, int valor){
207       int i;
208       /*
209   Como estou valiando a linha, entao percorrerei as colunas da linha
210   que sao no maximo 9 (de 0 a 8)*/
211       if(colunaTestar < 9)//Testo se o valor digitado se encontra na linha, ou
seja, nas colunas da linha{
212           if(valor != grade[linhaDigita][colunaTestar]){
213               if(!validaLinha(linhaDigita, colunaTestar+1, valor)){
214                       /*Este return so eh utilizado quando a recursividade
chegou ao fim
215   se chegou na ultima coluna da linha, e nao identificou numero repetido,
216   entao ele voltara recursivamente dando return zero*/
217                   return 0;
218               }else{
219                   /*Este return so eh utilizado quando a recursividade
encontra um numero
220   repetido, dai ele sai retornando verdadeiro*/
221                   return 1;
222               }
223           }else{
224               /*Este return nao faz exatamente a recursividade, mas pode estar
dentro dela,
225   exemplo: a recursividade executou 3 vezes e entrou neste return,
226   entao ela ira se desfazendo e dando returns verdadeiros*/
227               return 1;
228           }
229       else{
230           /*Este return so eh utilizado caso passe por todas colunas da linha
231   e nao encontrar nenhum valor repetido*/
232           return 0;
233       }
234   }
235
236   /*Essa recursão faz a mesma coisa da recursão acima, mas para a coluna*/
237   int validaColuna(int colunaDigita, int linhaTestar, int valor){
238       int i;
239       if(linhaTestar < 9){
240           if(valor != grade[linhaTestar][colunaDigita]){
241               if(!validaColuna(colunaDigita, linhaTestar+1, valor)){
242                   return 0;
243               }else{
244                   return 1;
245               }
246           }else{
247               return 1;
248           }
249       }else{
250           return 0;
```

```
251         }
252     }
```