```
/*
 *   Este programa pretende ser um exemplo do uso de conjuntos.
 *
 *   Tem como motivação a resolução de problemas do jogo Sudoku www.Sudoku.com
 *  do interesse da turma ARCIL0505 da ATEC.
 *
 *   O jogo consiste em colocar os dígitos 1 a 9 nos diversos espaços da tabela
 *  garantindo que existe uma e uma só ocorrência de cada dígito em cada coluna
 *  e em cada coluna. A tabela está ainda subdividida em 9 tabelas de 3x3 e em
 *  cada uma dessas sub-tabelas só pode haver também uma ocorrência dos dígitos
 *  de 1 a 9
 *
 *   Este programa está dimensionado para resolver problemas de 9 linhas e de
 *  9 colunas, mas não resolve o facto de se manter também uma ocorrência dos
 *  dígitos dentro de cada uma das sub-tabelas .... afinal, isto é para ser
 *  construído por todos ;)
 *
 *   O formador, Vitor Vaz da Silva. 2 Julho 2005
 *
*/
#include <stdio.h>
#include <stdlib.h>

#define MAX_COL    9 /* número máximo de colunas */
#define MAX_LIN    9 /* número máximo de linhas */
#define MAX_SUB    3 /* numero de linhas e colunas da sub-tabela */

/*   Para trabalhar com conjuntos, é necessário que cada elemento tenho um valor
 * diferente, e qualquer combinação de elementos tem de produzir também
 * combinações inequívocas.
 *   Deste modo cada elemento tem um peso diferente.
 *
*/

#define e0  0000 /*   000000000 */
#define e1  0001 /*   000000001 */
#define e2  0002 /*   000000010 */
#define e3  0004 /*   000000100 */
#define e4  0010 /*   000001000 */
#define e5  0020 /*   000010000 */
#define e6  0040 /*   000100000 */
#define e7  0100 /*   001000000 */
#define e8  0200 /*   010000000 */
#define e9  0400 /*   100000000 */
#define uu  0777 /*   111111111 */

/* definição de um novo tipo, o da tabela do jogo */
typedef int sudoku[MAX_LIN][MAX_COL];
```

/* declaração e iniciação da tabela original */

```
sudoku Mapa_teste = {
            {e1, 0, 0, 0, 0, 0, 0, 0, 0},
            {e2, 0, 0, 0, 0,e4, 0, 0, 0},
            { 0, 0, 0, 0, 0, 0, 0, 0, 0},
            { 0, 0,e3, 0, 0, 0, 0, 0, 0},
            { 0, 0, 0, 0,e5, 0, 0, 0, 0},
            { 0, 0,e9, 0, 0, 0, 0, 0, 0},
            { 0, 0, 0, 0, 0, 0,e7, 0, 0},
            { 0, 0, 0, 0, 0,e6,e8, 0, 0},
            { 0, 0, 0, 0, 0, 0, 0, 0, 0},
      };

sudoku Mapa_Site_Sudoku = {
            {e0,e6,e0, e1,e0,e4, e0,e5,e0},
            {e0,e0,e8, e3,e0,e5, e6,e0,e0},
            {e2,e0,e0, e0,e0,e0, e0,e0,e1},

            {e8,e0,e0, e4,e0,e7, e0,e0,e6},
            {e0,e0,e6, e0,e0,e0, e3,e0,e0},
            {e7,e0,e0, e9,e0,e1, e0,e0,e4},

            {e5,e0,e0, e0,e0,e0, e0,e0,e2},
            {e0,e0,e7, e2,e0,e6, e9,e0,e0},
            {e0,e4,e0, e5,e0,e8, e0,e7,e0},
      };

#define Mapa Mapa_Site_Sudoku     /* define o mapa a utilizar */

/* Conjuntos para conter os elementos disponíveis em cada linha e coluna*/
int ConjLin[MAX_LIN];
int ConjCol[MAX_COL];

/*
 * Valores devolvidos pelas funções.
 *
*/
#define OK            0   /* não há erro, solução encontrada */
#define NAO_SOLUCAO -1   /* indica que a escolha não leva a uma solução */

/*
 * Conjunto de macros utilizadas para interagir com os conjuntos.
 *
*/
#define existe(c, e)   (c & e)           /* verifica se e está incluído em c */
#define tira(c, e)     {c &= (~(e)) & uu ;}        /* subtrai e do conjunto c */
#define poe(c, e)      {c |= (e);}       /* adiciona e ao conjunto c */
#define limpa(c)       {c = 0;}          /* atribuir o conjunto vazio a c */
#define todos(c)       {c = uu;}         /* atribuir o universo a c */
#define intersec(c,d) (c & d)           /* intersecção entre dois conjuntos */
#define npe(n,e)       {for(e=0; n ;n--) e<<1;}   /* converte número p elemento */
```

```
/*
 * epn
 *
 *  Converte um elemento do conjunto para o seu correspondente numérico
 *
*/
int epn(int e)
{
int n;

  for(n=0; e ; n++) e= e >>1; /* converte elemento p número */
  return(n);
}

/*
 * proximo
 *
 *  Devolve o primeiro elemento do conjunto.
 *  Esta função parte do princípio que o conjunto a procurar não está vazio.
 *
*/
int proximo(int c)
{
int e;

  for(e=1; ! existe(c,e) ; e<<=1);
  return(e);
}

/*
 *  MostraConjuntos
 *
 *  Mostra os conjuntos das linhas e colunas mas indica só os números que estão
 * presentes para se poder compreender melhor o seu conteúdo.
 *  Esta função serve apenas para ajudar ao desenvolvimento do programa;
 * para teste. Não tem qualquer uso no cálculo do Mapa.
 *
*/
void MostraConjuntos()
{
int l,c;

  printf("L ");
  for(l=0 ; l<MAX_LIN ; l++) printf("%03o ",uu & ~ConjLin[l]);
  printf("\nC ");
  for(c=0 ; c<MAX_COL ; c++) printf("%03o ",uu & ~ConjCol[c]);
  printf("\n");
}
```

```
/*
 * Inicio
 *
 *  Iniciar todas as estruturas de dados globais (variáveis)
 *  Esta função não verifica se há algum erro na tabela inicial.
 *
*/
void Inicio()
{
int l,c;

   /* Iniciar os conjuntos de disponibilidade de elementos */
   for(l=0 ; l<MAX_LIN ; l++) todos(ConjLin[l]);
   for(c=0 ; c<MAX_COL ; c++) todos(ConjCol[c]);
   /* Retirar dos conjuntos de disponibilidade os dígitos utilizados */
   for(l=0 ; l<MAX_LIN ; l++)
   for(c=0 ; c<MAX_COL ; c++){
     tira(ConjLin[l], Mapa[l][c]);
     tira(ConjCol[c], Mapa[l][c]);
   }
}

/*
 * MostraMapa
 *
 *  Desenha o mapa com os seus números
 *  Para desenhar o mapa mostrando o elemento do conjunto dos dígitos, sugere-se
 *  não evocar a função epn, e mostrar em octal:  printf(" %03o", Mapa[l][c]);
 *
*/
void MostraMapa()
{
int l,c;

  for(l=0 ; l<MAX_LIN ; l++){
    if((l % MAX_SUB)==0) printf("\n");
    for(c=0 ; c<MAX_COL ; c++){
      if((c % MAX_SUB)==0) printf("  ");
      printf(" %d", epn(Mapa[l][c]));
    }
    putchar('\n');
  }
}
```

```
/*
 * AtribuiElemento
 *
 *  Esta é a função que atribui um novo número ao Mapa de acordo com as regras
 * A função é evocada recursivamente procurando assim as diversas hipóteses
 * A variável res contém as diversas hipóteses para a célula (l,c) em questão
 * e procura
 *
*/
AtribuiElemento(sudoku tab)
{
int l,c,e,res;

/* Esta macro Passo ajuda para verificar o modo como funciona este algoritmo. É
 * utilizada apenas para teste e não tem qualquer influência sob o resultado.
*/
#define Passo {  MostraMapa(); MostraConjuntos(); \
    printf("l %d c %d res %03o e %03o tab[l][c] %d\n",l, c, res,e,tab[l][c] ); \
    getchar();}

  for(l=0 ; l<MAX_LIN ; l++){
    for(c=0 ; c<MAX_COL ; c++){
      if(tab[l][c]!=0) continue;       /* esta célula não tem elemento seguinte ... */
      res=intersec(ConjLin[l], ConjCol[c]);   /* os elementos possíveis */
      if (res==0) return(NAO_SOLUCAO);      /* se vazio então corta a recursividade */
      while (res) {                    /* res contém os números a testar */
        e=proximo(res);           /* venha o primeiro (seguinte) número */
        tira(res, e);             /* retira-se de res para não ser testado novamente */
        tab[l][c]=e;              /* atribui-se o número à célula (l,c) do Mapa */
        tira(ConjLin[l], e);      /* retira-se o elemento do conjunto da linha */
        tira(ConjCol[c], e);      /* retira-se o elemento do conjunto da coluna */
        if(AtribuiElemento(tab)!= OK){   /* por aqui não se chega lá, por isso */
          tab[l][c]=0;                   /* marcar a posição como livre */
          poe(ConjLin[l], e);            /* repor o número na linha */
          poe(ConjCol[c], e);            /* e na coluna */
        }
        else return(OK);  /* se não há erro então encontrou-se a solução */
    /* testar com o próximo número */
      }
      if(res==0) return(NAO_SOLUCAO);  /* se vazio então corta a recursividade */
    }
  }
  return(OK);  /* Se já está tudo atribuído, é porque foi encontrada a solução */
}
```

```
/*
 * main
 *
 *
 *
*/
int main(int argc, char *argv[])
{
  Inicio();
  printf("  Calculo de soluções para o jogo SuDoKu. http://sudoku.com\n");
  printf("Esta versão v1 não está completa propositadamente: \n\n");
  printf("  Sugestão: - Completar o algoritmo para que não haja repetições \n");
  printf("            nos diversos grupos de 9\n");
  printf("          - Permitir ao utilizador introduzir um novo mapa \n");
  printf("            e verificar se os valores estão correctos. \n");
  printf("          - Gerar um novo Mapa aleatoriamente. O utilizador \n");
  printf("            indica quantos números quer inicialmente no Mapa \n");
  printf("  Turma ARCIL 0505 - ATEC, Vitor Vaz da Silva \n\n");
  printf("Mapa Inicial: \n");
  MostraMapa();                                          /* Mostra o Mapa */
  printf("Prima <enter> para começar ...  ");
  getch();
  if(AtribuiElemento(Mapa)==OK){                         /* Procura uma Solução */
    printf("Foi encontrada uma solução: \n");
    MostraMapa();                                        /* Mostra o Mapa */
  }
  else printf("\n\n Não foi possível encontrar uma solução ...\n");
  system("PAUSE");
  return 0;
}
```