```
#include "io430.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>


// Insere o cursor na primeira linha

#define CURSOR_PRIMEIRA_LINHA 128   // 100000000 set display address// Insere o cursor na segunda linha

#define CURSOR_SEGUNDA_LINHA 192    // 110000000 set display address// Coloca o cursor piscando#define SHOW_CURSOR_PISCANDO 15

#define CLEAR_DISPLAY  1 // 01  clear display and cursor home

#define CLEAR_CURSOR  3 // 11  clear display and cursor home// Esconde o cursor

#define HIDE_CURSOR  12  // 1100 display off and cursor underline off and cursor blink off// Mostra o cursor apenas o underline _

#define SHOW_CURSOR  14  // 1110 display oN and cursor underline ON// Configura o display para 2 linhas, 8 bits e 5x7 - function set

#define DISPLAY_2LINHAS_8BITS_5X7 56 // Liga o display com o cursor desligado

#define LIGAR_DISPLAY_CURSOR_DESLIGADO  12 // Liga o display com o cursor desligado - 1100 display off and cursor underline off and cursor blink off// codigo da flecha para a direita

#define FLECHA_ESQUERDA 127 // codigo a flecha para a direita

#define FLECHA_DIREITA 231

#define SHIFT_DIREITA 28

#define SHIFT_ESQUERDA 24

#define BOAS_VINDAS "OI!! Bem vindo ao VinBin."

#define RECORDES "Recordes"

#define JOGAR "Jogar"

#define PAUSE_COMANDO 1

#define PAUSE_FRASE 2

#define CIMA 1
```

```
#define DIREITA 2

#define BAIXO 4

#define ESQUERDA 8

// Coloca o cursor piscando

#define SHOW_CURSOR_PISCANDO 15


struct pontos {

  // Nome da carta

  char nome[9];

  int pontos;

  int vazio;

};


struct cartaBaralho {

  // Nome da carta

  char *nome;// Indica o naipe  // Naipe: Ouros, Copas , Espadas, Paus

  char *naipe;  // Valor em pontos do baralho  // Valor: númericas: os valores serãos seu snúmeros,  Valete (J), Dama (Q) e Reis (K) terão o valor de 10, Ás(A) terá valor de 11

  int valor;  // Flag que indica que esta no baralho  // Flag que indica se foi retirado a carta do baralho ou não. 0 = continua no baralho, 1 = Retirada do baralho

  int noBaralho;

};


/*****************************************/

// Funções do LCD

void Print(char* str);

void EnviarPausa();

void enviarComando(int caracter, int comando);
```

```
void Display16x2();

void clear();

void SetPosition(int linha, int coluna);

void shiftEsquerda();

void shiftDireta();


/*****************************************/

void configurarDisplay();

void imprimirBoasVindas();

void mostrarMenu();

void mostrarRecorde();

void jogar();

int lerComando();

int opcaoMenu();

void ShowCursorPiscando();

void intToString(int num, char *numero);

void mostrarRecorde();

int iniciarJogo();

void iniciarExecucao();

int recuperarCarta();

void montaPlacar();

void mensagemFinalJogo(int tipo);

int cadastrarRecorde();

void printAlfabeto(int NrAlfabeto, char * nome);

void inserirRecorde(char * nome);

void ShowCursorSublinhado();
```

```
typedef struct cartaBaralho carta;

carta baralho[52];

struct pontos arrayRecordes[10];

int totalMaquina;

int totalJogador;

char alfabeto[] = "ABCDEFGHIJKLMNOPQRSTWUYXZ";

char alfabeto1[] = "ABCDEFGHIJKLMNOP";

char alfabeto2[] = "QRSTWUYXZ";

int _uPos = 0;

int _uTam = 0;

char _uNome[9];



/*********************************cartas*************************************
****/

// Função que monta o veto de cartas


/*********************************cartas*************************************
****/


// Função que monta o veto de cartas


carta As_O;


carta As_C;


carta As_E;
```

carta As_P;

carta Dois_O;

carta Dois_C;

carta Dois_E;

carta Dois_P;

carta Tres_O;

carta Tres_C;

carta Tres_E;

carta Tres_P;

carta Quatro_O;

carta Quatro_C;

carta Quatro_E;

carta Quatro_P;

carta Cinco_O;

carta Cinco_C;

carta Cinco_E;

carta Cinco_P;

carta Seis_O;

carta Seis_C;

carta Seis_E;

carta Seis_P;

carta Sete_O;

carta Sete_C;

carta Sete_E;

carta Sete_P;

carta Oito_O;

carta Oito_C;

carta Oito_E;

carta Oito_P;

carta Nove_O;

carta Nove_C;

carta Nove_E;

carta Nove_P;

carta Dez_O;

carta Dez_C;

carta Dez_E;

carta Dez_P;

carta Valete_O;

carta Valete_C;

carta Valete_E;

carta Valete_P;

```
carta Dama_O;
  carta Dama_C;
  carta Dama_E;
  carta Dama_P;

carta Reis_O;
  carta Reis_C;
  carta Reis_E;
  carta Reis_P;

char  nome1[3] = "As";

char  nome2[2] = "2";

char  nome3[2] = "3";

char  nome4[2] = "4";

char  nome5[2] = "5";

char  nome6[2] = "6";

char  nome7[2] = "7";
```

```
char  nome8[2] = "8";

char  nome9[2] = "9";

char  nome10[3] = "10";

char  nome11[6] = "Valet";

char  nome12[5] = "Dama";

char  nome13[5] = "Reis";



char  naipe1[6] = "Ouros";

char  naipe2[5] = "Copas";

char  naipe3[6] = "Espad";

char  naipe4[6] = "Paus";

/*
  FUNÇÕES AUXILIARES PARA EVITAR REPETIÇÃO DE CÓDIGO
*/

// Função necessária para criar uma pausa para enviar o comando para o display
```

```
void EnviarPausa() {
  int i;
  // for de 200 para criar uma palsa para enviar o comando para o lcd
  for (i = 0; i < 200; i++);
}

// Função auxiliar que indica quando é para enviar comando ou caracter
// caracter = 1 o envio será caracteres, caracter = 0 séra enviado um comando
// Comando = é o valor de um comando ou caracter
void enviarComando(int caracter, int comando) {
  // P2OUT indica se será um comando ou caracter
  // 1 = Comando
  if (caracter == 1)
    P2OUT = 0X40; // Seta o RS como um para enviar caracter, P2.6 e enable como zero, 1000000
  else
    P2OUT = 0X00; // Seta o RS como zero e Enable como zero


  P5OUT = comando; // Seta o comando na porta 5
  EnviarPausa();  // Envia a pausa para inserir os comandos na porta ou caracter

  // Se for enviar caracteres mantem RS em 1
   if (caracter == 1)
    P2OUT = 0xC0;        // Seta enable como 1 para iniciar o envio do comando e mante o rs em 1
  // Se for enviar um comando mantem o RS em 0
  else
```

```
   P2OUT =  0x80;       // Seta enable como 1 iniciar e matem o rs como zero para enviar caracteres


  // Coloca o enable em zero para terminar o envio do comando

  if (caracter == 1)

   P2OUT = 0x40;       // Seta enable como 0 para finalizar o envio do comando e mante o rs em um

  // Se for enviar caracteres mantem RS em 0

  else

   P2OUT =  0x00;       // Seta enable como 0 para finalizar e mantem o rs como zero para enviar comando


}


/*

  Funções requeridas da atividade

*/


// configura o display para duas linhas sem cursor;

void Display16x2() {

  enviarComando(0, LIGAR_DISPLAY_CURSOR_DESLIGADO);

  enviarComando(0, DISPLAY_2LINHAS_8BITS_5X7);

}


// imprime a string str no cursor.

void Print(char* str) {

    int i;

  for (i = 0; str[i] != '\0'; i++)

   enviarComando(1, str[i]);
```

```
}

// limpa o display e posiciona o cursor na 1a posição;
void clear() {
  // Coloca o RS em zero para enviar comandos
  // Envia o comando
  enviarComando(0, CLEAR_DISPLAY);
  enviarComando(0, CLEAR_CURSOR);
}

// apaga o cursor;
void HideCursor() {
  // Coloca o RS em zero para enviar comandos
  // Envia o comando
  enviarComando(0, HIDE_CURSOR);
}

// posiciona o cursor do display em uma linha/coluna;
void SetPosition(int linha, int coluna) {
  enviarComando(0, coluna + linha);
}

// exibe o cursor;
void ShowCursor() {
   // Coloca o RS em zero para enviar comandos
  // Envia o comando
  enviarComando(0, SHOW_CURSOR);
```

```
}

// exibe o cursor;
void ShowCursorSublinhado() {
   // Coloca o RS em zero para enviar comandos
  // Envia o comando
  enviarComando(0, SHOW_CURSOR);
  enviarComando(0, SHOW_CURSOR);
}

void ShowCursorPiscando() {
   // Coloca o RS em zero para enviar comandos
  // Envia o comando
  enviarComando(0, SHOW_CURSOR_PISCANDO);
}



int lerComando(){
  int comando = 0;
  while (comando == 0)
    comando =  P3IN;
return comando;
}

int montarCartas() {

  As_O.nome = nome1;
```

```
As_O.naipe = naipe1;

As_O.valor = 11;

As_O.noBaralho = 0;


baralho[0] = As_O;


As_C.nome = nome1;

As_C.naipe = naipe2;

As_C.valor = 11;


As_C.noBaralho = 0;


baralho[1] = As_C;



As_E.nome = nome1;


As_E.naipe = naipe3;


As_E.valor = 11;


As_E.noBaralho = 0;


baralho[2] = As_E;



As_P.nome = nome1;
```

```
As_P.naipe = naipe4;

As_P.valor = 11;

As_P.noBaralho = 0;

baralho[3] = As_P;



Dois_O.nome = nome2;

Dois_O.naipe = naipe1;

Dois_O.valor = 2;

Dois_O.noBaralho = 0;

baralho[4] = Dois_O;



Dois_C.nome = nome2;

Dois_C.naipe = naipe2;

Dois_C.valor = 2;
```

```
Dois_C.noBaralho = 0;

baralho[5] = Dois_C;



Dois_E.nome = nome2;

Dois_E.naipe = naipe3;

Dois_E.valor = 2;

Dois_E.noBaralho = 0;

baralho[6] = Dois_E;



Dois_P.nome = nome2;

Dois_P.naipe = naipe4;

Dois_P.valor = 2;

Dois_P.noBaralho = 0;

baralho[7] = Dois_P;
```

```
Tres_O.nome = nome3;

Tres_O.naipe = naipe1;

Tres_O.valor = 3;

Tres_O.noBaralho = 0;

baralho[8] = Tres_O;



Tres_C.nome = nome3;

Tres_C.naipe = naipe3;

Tres_C.valor = 3;

Tres_C.noBaralho = 0;

baralho[9] = Tres_C;



Tres_E.nome = nome3;

Tres_E.naipe = naipe3;

Tres_E.valor = 3;
```

```
Tres_E.noBaralho = 0;

baralho[10] = Tres_E;



Tres_P.nome = nome3;

Tres_P.naipe = naipe4;

Tres_P.valor = 3;

Tres_P.noBaralho = 0;

baralho[11] = Tres_P;



Quatro_O.nome = nome4;

Quatro_O.naipe = naipe1;

Quatro_O.valor = 4;

Quatro_O.noBaralho = 0;

baralho[12] = Quatro_O;
```

```
Quatro_C.nome = nome4;

Quatro_C.naipe = naipe3;

Quatro_C.valor = 4;

Quatro_C.noBaralho = 0;

baralho[13] = Quatro_C;



Quatro_E.nome = nome4;

Quatro_E.naipe = naipe3;

Quatro_E.valor = 4;

Quatro_E.noBaralho = 0;

baralho[14] = Quatro_E;



Quatro_P.nome = nome4;

Quatro_P.naipe = naipe4;
```

```
Quatro_P.valor = 4;

Quatro_P.noBaralho = 0;

baralho[15] = Quatro_P;



Cinco_O.nome = nome5;

Cinco_O.naipe = naipe1;

Cinco_O.valor = 5;

Cinco_O.noBaralho = 0;

baralho[16] = Cinco_O;



Cinco_C.nome = nome5;

Cinco_C.naipe = naipe2;

Cinco_C.valor = 5;

Cinco_C.noBaralho = 0;

baralho[17] = Cinco_C;
```

```
Cinco_E.nome = nome5;

Cinco_E.naipe = naipe3;

Cinco_E.valor = 5;

Cinco_E.noBaralho = 0;

baralho[18] = Cinco_E;


Cinco_P.nome = nome5;

Cinco_P.naipe = naipe4;

Cinco_P.valor = 5;

Cinco_P.noBaralho = 0;

baralho[19] = Cinco_P;


Seis_O.nome = nome6;

Seis_O.naipe = naipe1;
```

```
Seis_O.valor = 6;

Seis_O.noBaralho = 0;

baralho[20] = Seis_O;



Seis_C.nome = nome6;

Seis_C.naipe = naipe2;

Seis_C.valor = 6;

Seis_C.noBaralho = 0;

baralho[21] = Seis_C;



 Seis_E.nome = nome6;

Seis_E.naipe = naipe3;

Seis_E.valor = 6;

Seis_E.noBaralho = 0;
```

```
baralho[22] = Seis_E;



Seis_P.nome = nome6;

Seis_P.naipe = naipe4;

Seis_P.valor = 6;

Seis_P.noBaralho = 0;

baralho[23] = Seis_P;



Sete_O.nome = nome7;

Sete_O.naipe = naipe1;

Sete_O.valor = 7;

Sete_O.noBaralho = 0;

baralho[24] = Sete_O;



Sete_C.nome = nome7;
```

```
Sete_C.naipe = naipe2;

Sete_C.valor = 7;

Sete_C.noBaralho = 0;

baralho[25] = Sete_C;



Sete_E.nome = nome7;

Sete_E.naipe = naipe3;

Sete_E.valor = 7;
 Sete_E.noBaralho = 0;
 baralho[26] = Sete_E;

 Sete_P.nome = nome7;
 Sete_P.naipe = naipe4;
 Sete_P.valor = 7;

Sete_P.noBaralho = 0;

baralho[27] = Sete_P;

 Oito_O.nome = nome8;
 Oito_O.naipe = naipe1;
```

```
 Oito_O.valor = 8;
 Oito_O.noBaralho = 0;

baralho[28] = Oito_O;

 Oito_C.nome = nome8;
 Oito_C.naipe = naipe2;
 Oito_C.valor = 8;
 Oito_C.noBaralho = 0;

baralho[29] = Oito_C;

 Oito_E.nome = nome8;
 Oito_E.naipe = naipe3;
 Oito_E.valor = 8;
 Oito_E.noBaralho = 0;

baralho[30] = Oito_E;

 Oito_P.nome = nome8;
 Oito_P.naipe = naipe4;
 Oito_P.valor = 8;
 Oito_P.noBaralho = 0;

baralho[31] = Oito_P;

 Nove_O.nome = nome9;
```

```
 Nove_O.naipe = naipe1;
 Nove_O.valor = 9;
 Nove_O.noBaralho = 0;

baralho[32] = Nove_O;

 Nove_C.nome = nome9;
 Nove_C.naipe = naipe2;
 Nove_C.valor = 9;
 Nove_C.noBaralho = 0;

baralho[33] = Nove_C;

 Nove_E.nome = nome9;
 Nove_E.naipe = naipe3;
 Nove_E.valor = 9;
 Nove_E.noBaralho = 0;

baralho[34] = Nove_E;

 Nove_P.nome = nome9;
 Nove_P.naipe = naipe4;
 Nove_P.valor = 9;
 Nove_P.noBaralho = 0;

baralho[35] = Nove_P;
```

```
 Dez_O.nome = nome10;
 Dez_O.naipe = naipe1;
 Dez_O.valor = 10;
 Dez_O.noBaralho = 0;

baralho[36] = Dez_O;

 Dez_C.nome = nome10;
 Dez_C.naipe = naipe2;
 Dez_C.valor = 10;
 Dez_C.noBaralho = 0;

baralho[37] = Dez_C;

 Dez_E.nome = nome10;
 Dez_E.naipe = naipe3;
 Dez_E.valor = 10;
 Dez_E.noBaralho = 0;

baralho[38] = Dez_E;

 Dez_P.nome = nome10;
 Dez_P.naipe = naipe4;
 Dez_P.valor = 10;
 Dez_P.noBaralho = 0;

baralho[39] = Dez_P;
```

```
 Valete_O.nome = nome11;
 Valete_O.naipe = naipe1;
 Valete_O.valor = 10;
 Valete_O.noBaralho = 0;

baralho[40] = Valete_O;

 Valete_C.nome = nome11;
 Valete_C.naipe = naipe2;
 Valete_C.valor = 10;
 Valete_C.noBaralho = 0;

baralho[41] = Valete_C;

 Valete_E.nome = nome11;
 Valete_E.naipe = naipe3;
 Valete_E.valor = 10;
 Valete_E.noBaralho = 0;

baralho[42] = Valete_E;

 Valete_P.nome = nome11;
 Valete_P.naipe = naipe4;
 Valete_P.valor = 10;
 Valete_P.noBaralho = 0;
```

```
baralho[43] = Valete_P;

 Dama_O.nome = nome11;
 Dama_O.naipe = naipe1;
 Dama_O.valor = 10;
 Dama_O.noBaralho = 0;

baralho[44] = Dama_O;

 Dama_C.nome = nome11;
 Dama_C.naipe = naipe2;
 Dama_C.valor = 10;
 Dama_C.noBaralho = 0;

baralho[45] = Dama_C;

 Dama_E.nome = nome11;
 Dama_E.naipe = naipe3;
 Dama_E.valor = 10;
 Dama_E.noBaralho = 0;

baralho[46] = Dama_E;

 Dama_P.nome = nome11;
 Dama_P.naipe = naipe4;
 Dama_P.valor = 10;
 Dama_P.noBaralho = 0;
```

```
baralho[47] = Dama_P;

 Reis_O.nome = nome12;
 Reis_O.naipe = naipe1;
 Reis_O.valor = 10;
 Reis_O.noBaralho = 0;

baralho[48] = Reis_O;

 Reis_C.nome = nome12;
 Reis_C.naipe = naipe2;
 Reis_C.valor = 10;
 Reis_C.noBaralho = 0;

baralho[49] = Reis_C;

 Reis_E.nome = nome12;
 Reis_E.naipe = naipe3;
 Reis_E.valor = 10;
 Reis_E.noBaralho = 0;

baralho[50] = Reis_E;

 Reis_P.nome = nome12;
 Reis_P.naipe = naipe4;
 Reis_P.valor = 10;
```

```
   Reis_P.noBaralho = 0;

  baralho[51] = Reis_P;
  return 0;
}

void intToString(int num, char *numero) {
  if (num > 9)
    sprintf(numero, "%d", num);
  else
    sprintf(numero, "%d", num);
}

int recuperarCarta() {
    // Variavel que indica a posição da carta que será retirada
  int cartaRetirada;

  // Flag que indica se sera preciso tirar outra carta ou não

  int outraCarta = 0;

  while (outraCarta == 0) {

    cartaRetirada = ((rand()) % 52);

    // Verifica se a carta já foi retirada
```

```
    if (baralho[cartaRetirada].noBaralho == 0) {

      outraCarta = 1;
      // Seta a carta como retirada do abrarlho

      baralho[cartaRetirada].noBaralho = 1;

    }

  }

  return cartaRetirada;

}

void imprimirBoasVindas() {
  // Procedimento que imprimi as boas vindas
  Print(BOAS_VINDAS);
}

void mostrarMenu() {
  clear();
  SetPosition(CURSOR_PRIMEIRA_LINHA, 1);
  Print(JOGAR);
  SetPosition(CURSOR_SEGUNDA_LINHA, 1);
  Print(RECORDES);
  ShowCursorPiscando();
```

```
}

int opcaoMenu() {
  int _rOpcao = CIMA;
  int _rPos = 0;
  while (_rOpcao == CIMA || _rOpcao == BAIXO) {
    _rOpcao = lerComando();
    if (_rOpcao == CIMA && _rPos != 0) {
      SetPosition(CURSOR_SEGUNDA_LINHA, 0);
      _rPos = 0;
    }
    else if (_rOpcao == BAIXO && _rPos != 1) {
      SetPosition(CURSOR_SEGUNDA_LINHA, 0);
      _rPos = 1;
    }
    else if (_rOpcao == DIREITA || _rOpcao == ESQUERDA) {
      return _rPos;
    }
  }
  return 0;
}

void mostrarRecorde() {
    int _rComando = 0;
    int _rPos = 0;
    int _rUltimo = 0;
    char Nm[] = "Nm.";
```

```
char Pt[] = "Pt.";
char _rNumero[2];
char NaoRecordes[] = "Nao tem recordes.";
struct pontos jogador;

// Se a primeira posição for vazia não sera impressa o record
if (arrayRecordes[0].vazio == 0) {
   Print(NaoRecordes);
   lerComando();
   return;
}
clear();
while (1 == 1) {
   // Limpa o lcd
   jogador = arrayRecordes[_rPos];
   if (jogador.vazio != 0) {
      _rUltimo = 0;
      clear();
      SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
      Print(Nm);
      SetPosition(CURSOR_PRIMEIRA_LINHA, 4);
      Print(jogador.nome);
      SetPosition(CURSOR_PRIMEIRA_LINHA, 13);
      memset(_rNumero, '\0', 2);
      Print(Pt);
      intToString(jogador.pontos, _rNumero);
      Print(_rNumero);
```

```
}
else {
   _rPos -= 2;
   _rUltimo = 1;
}
if (_rUltimo == 0) {
   // Verifica a segunda posição
      jogador = arrayRecordes[_rPos + 1];
   if (jogador.vazio != 0) {
      SetPosition(CURSOR_SEGUNDA_LINHA, 0);
      Print(Nm);
      SetPosition(CURSOR_SEGUNDA_LINHA, 4);
      Print(jogador.nome);
      SetPosition(CURSOR_SEGUNDA_LINHA, 13);
      memset(_rNumero, '\0', 2);
      Print(Pt);
      intToString(jogador.pontos, _rNumero);
      Print(_rNumero);
   }
}
_rComando = lerComando();
// Se o comando for esquerda ou direita retornara
if (_rComando == ESQUERDA || _rComando == DIREITA)
   return;
if (_rComando == BAIXO && _rPos < 10)
 _rPos = _rPos + 2;
else if (_rComando == CIMA && _rPos >= 2)
```

```
            _rPos = _rPos - 2;
        }
    }

    void jogar() {
      char mensagem[] = "O Jogo vai comecar!!";
      clear();
      Print(mensagem);
      lerComando();
      clear();
      enviarComando(1, FLECHA_DIREITA);
      Print("= pedir nova carta");
      lerComando();
      clear();
      enviarComando(1, FLECHA_ESQUERDA);
      Print("= desistir do jogo");
      lerComando();
      montarCartas();
      iniciarJogo(1, -3);
    }
    int iniciarJogo() {
        char mensagem[16];
        int _rJogador = 1;
        int _rComando = -3;
        int CartaRecuperada;
        while (1 == 1) {
            clear();
```

```
        montaPlacar();

        CartaRecuperada = recuperarCarta();

        if (_rJogador == 1) {

           memset(mensagem, '\0', 16);

           sprintf(mensagem, "Vc=%s %s Pt.%d", baralho[CartaRecuperada].nome,
baralho[CartaRecuperada].naipe, baralho[CartaRecuperada].valor);

           Print(mensagem);

           totalJogador += baralho[CartaRecuperada].valor;

           if (totalJogador > 21) {

              return 1;

           }

           else {

              _rComando = lerComando();

              while(_rComando == CIMA &&  _rComando == BAIXO) {

                 _rComando = lerComando();

              }

              _rJogador = 0;

           }

        }

        else {

           // Pediu nova carda

           if (_rComando == DIREITA) {

              memset(mensagem, '\0', 16);

              sprintf(mensagem, "EU=%s %s Pt.%d", baralho[CartaRecuperada].nome,
baralho[CartaRecuperada].naipe, baralho[CartaRecuperada].valor);

              Print(mensagem);

              totalMaquina += baralho[CartaRecuperada].valor;

              if (totalMaquina > 21) {
```

```
            return 2;
        }
        else {
            _rComando = lerComando();
        while(_rComando == CIMA &&  _rComando  == BAIXO) {
            _rComando = lerComando();
        }
        _rJogador = 1;
        }
    }
    // Desistiu
    else {
        if (totalMaquina > totalJogador) {
            return 1;
        }
        else if (totalMaquina < totalJogador) {
            memset(mensagem, '\0', 16);
            sprintf(mensagem, "EU=%s %s %d", baralho[CartaRecuperada].nome,
baralho[CartaRecuperada].naipe, baralho[CartaRecuperada].valor);
            Print(mensagem);
            totalMaquina += baralho[CartaRecuperada].valor;
            if (totalMaquina > 21) {
                return 2;
            }
            else if (totalMaquina > totalJogador) {
                return 1;
            }
            else
```

```
                return 2;
            }
        }
    }

}

void mensagemFinalJogo(int tipo) {
    clear();
    char mensagem[16];
    switch (tipo) {
        case 1:
            memset(mensagem, '\0', 16);
            SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
            Print("Voce perdeu");
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            sprintf(mensagem, "Seus pontos %d", totalJogador);
            Print(mensagem);
            lerComando();
            memset(mensagem, '\0', 16);
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            Print("Eu ganhei hahaha");
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            sprintf(mensagem, "Meus pontos %d", totalMaquina);
            Print(mensagem);
```

```
            lerComando();
        break;
        case 2:
            memset(mensagem, '\0', 16);
            SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
            Print("Voce ganhou prbns");
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            sprintf(mensagem, "Seus pontos %d", totalJogador);
            Print(mensagem);
            lerComando();
            memset(mensagem, '\0', 16);
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            Print("Eu perdi bhuaaa");
            SetPosition(CURSOR_SEGUNDA_LINHA, 0);
            sprintf(mensagem, "Meus pontos %d", totalMaquina);
            Print(mensagem);
            lerComando();
        break;

    }
}



int cadastrarRecorde() {
    int comando;
    _uTam = 0;
    int _rSair = 0;
```

```
clear();
Print("Digite seu nome.");


lerComando();
clear();
memset(_uNome, '\0', 9);
ShowCursorSublinhado();
SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
printAlfabeto(1, _uNome);
 while (_rSair == 0) {
   comando = lerComando();
   if (comando == DIREITA && _uPos == 15) {
      printAlfabeto(2, _uNome);
      _uPos ++;
   }
   else if (comando == DIREITA && _uPos < 25) {
      _uPos ++;
      shiftDireta();
   }
   else if (comando == ESQUERDA && _uPos == 16) {
      _uPos --;
      printAlfabeto(1, _uNome);
   }
   else if (comando == ESQUERDA && _uPos > 0) {
      _uPos --;
      shiftEsquerda();
   }
```

```
        else if (comando == CIMA) {
            _uNome[_uTam] = alfabeto[_uPos];
            _uTam ++;
            if (_uPos > 15)
                printAlfabeto(2, _uNome);
            else
                printAlfabeto(1, _uNome);
        }
        if (comando == BAIXO || _uTam == 8)
            _rSair = 1;
    }
    return 0;
}

void inserirRecorde(char * nome) {
    struct pontos aux0;
    struct pontos aux1;
    int i;
    memset(aux0.nome, '\0', 9);
    sprintf(aux0.nome, "%s", nome);
    aux0.pontos = totalJogador;
    aux0.vazio = 1;
    if(arrayRecordes[0].vazio == 0) {
        arrayRecordes[0] = aux0;
        return;
    }
    else {
```

```
      for (i = 0; i < 10 ; i++) {
         if (aux0.pontos >= arrayRecordes[i].pontos) {
            aux1 = arrayRecordes[i];
            arrayRecordes[i] = aux0;
            aux0 = aux1;
         }
      }

   }
   return;
}

void shiftEsquerda(){
  enviarComando(0, SHIFT_ESQUERDA);
}

void shiftDireta(){
  enviarComando(0, SHIFT_DIREITA);
}

void printAlfabeto(int NrAlfabeto, char * nome) {
   clear();
   if (NrAlfabeto == 1)
      Print(alfabeto1);
   else
      Print(alfabeto2);
   SetPosition(CURSOR_SEGUNDA_LINHA, 0);
```

```
    Print(nome);
    SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
    ShowCursorSublinhado();
    return;
}

void montaPlacar() {
  char placar[16];
  SetPosition(CURSOR_SEGUNDA_LINHA, 0);
  sprintf(placar, "Maq:%d  Voce:%d", totalMaquina, totalJogador);
  Print(placar);
  SetPosition(CURSOR_PRIMEIRA_LINHA, 0);
}



int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  // COnfigura as portas como saída P5 e P2
  P5DIR = 0xFF;
  P2DIR = 0xFF;
  // cofigura a porta 3 como entrada
  P3DIR = 0X00;
  // Configura o Display
  // configura o display para duas linhas sem cursor;
  iniciarExecucao();
```

```
  return 0;
}



void iniciarExecucao() {
  int _rOpcao;
  int _rGanhador;
  int _rNovoJogo = 1;
  Display16x2();
  while (1 == 1) {
    if (_rNovoJogo == 1) {
      // Limpa o Display
     clear();
      // imprimi as boas vindas
      imprimirBoasVindas();
      _rNovoJogo = 0;
      lerComando();
      totalJogador = 0;
      totalMaquina = 0;
    }
    // Mostra o meu
   mostrarMenu();
    _rOpcao = opcaoMenu();
    if (_rOpcao == 1)
      mostrarRecorde();
    else {
      jogar();
```

```
    _rGanhador = iniciarJogo();
    mensagemFinalJogo(_rGanhador);
    if (_rGanhador == 2) {
      cadastrarRecorde();
      inserirRecorde(_uNome);
    }
    _rNovoJogo = 1;
   }
  }
}
```