



**UNIDADE CURRICULAR:** Introdução à Programação

**CÓDIGO:** 21173

**DOCENTE:** José Coelho

**NOME:** Carlos Miguel Faria Martins da Costa

**N.º DE ESTUDANTE:** 2203524

**CURSO:** Licenciatura em Engenharia Informática

Autoavaliação:

Critérios	Alínea A	Alínea B	Alínea C	Alínea D
Funcionalidade	sim	sim	sim	sim
Qualidade	Efetuado			
Relatório dimensão / forma	De acordo com a regra			
Relatório conteúdo	De acordo com os critérios			
Relatório testes	Realizados como pedido			

Autoavaliação de acordo com os critérios de correção: +0.1 na nota do e-fólio  
Ver critérios de correção no espaço da UC ([aqui](#))

## **Relatório**

### **Alínea A:**

Para a implementação deste exercício e após definir 3 vetores, optei por 3 laços 'for' de forma a criar todas as combinações possíveis pedidas no exercício.

Tive certa dificuldade foi em encontrar solução para obter os índices corretos de forma a poder aceder aos 3 vetores tendo em conta o número inserido pelo jogador 'vCodigo'. A minha dificuldade foi em encontrar a fórmula para calcular 'j' 'índice para numeros', no entanto deparei-me que ao recorrer ao resto da divisão de 'vCodigo' por 16 e dividindo por 4 conseguia assim distribuir os 4 números possíveis.

Foram realizados testes sendo o output de acordo com o esperado, ver anexos Alínea A.

No teste de qualidade verifico '#EQ4: Nomes sem significado', após várias tentativas não consegui corrigir o erro.

### **Alínea B:**

Neste exercício, a função 'randaux' que gera números aleatórios e a função 'Baralhar' utilizada para baralhar as cartas são facultadas no enunciado.

Na função 'main' é-nos facultado uma parte do código nomeadamente 'desperdício', que dependentemente do número inserido pelo utilizador, servirá para criar diferentes baralhos nos casos teste – como expresso no enunciado.

Recorrendo aos vetores do exercício anterior, procurei depois declarar e preencher o vetor 'ordem' com os valores das posições de 0 a 63. Em seguida para criar uma disposição aleatória das cartas recorri à função 'Baralhar'.

Depois passei à fase de imprimir as cartas em formato de matriz. Penso que foi a parte que me deu mais trabalho pois não tinha percebido a ordem de impressão; após algumas tentativas e erro, utilizei os laços 'for' para iniciar um

laço externo para imprimir as cartas em linhas e outro laço interno para a parte das colunas. Em seguida, ao utilizar o vetor 'ordem' consegui assim ter o índice que corresponde à carta a ser impressa. Para finalizar, recorri à mesma fórmula utilizada no exercício anterior para calcular os índices dos vetores 'operacoes', 'numeros' e 'letras' com base no índice da carta.

Também devo referir que recorri às constantes 'NUM\_CARTAS 64' e 'CARTAS\_POR\_LINHA 8' para garantir um baralho com 64 cartas e que seria impresso 8 cartas por cada linha.

Os casos teste e casos teste extra efetuados tiveram o output esperado conforme se pode ver nos anexos Alínea B.

Relativamente ao teste de qualidade observo '#EQ4: Nomes sem significado', mas como estas variáveis da linha 12 são fornecidas pelo enunciado optei por não modificar.

### **Alínea C:**

Nesta alínea temos uma continuidade do exercício anterior aonde podemos servir-nos do código desenvolvido até aqui, nomeadamente até à parte aonde se utiliza a função 'Baralhar' para criar uma disposição aleatória das cartas.

O desafio deste exercício consistiria em revelar na matriz somente a sequência de 4 números inteiros positivos pedidos pelo utilizador e caso uma posição pedida caísse fora da matriz 8 x 8, não seria processada mais nada.

Utilizei assim, o vetor 'posicoes' e o laço 'for' para guardar as posições pedidas pelo utilizador. Depois para verificar se a posição se encontra fora do intervalo da matriz (11 - 88), recorro ao ciclo 'for' e condicional 'if' para validar as posições.

Recorri à mesma sequência de código desenvolvida no exercício anterior para imprimir as cartas por ordem, começando depois o maior desafio neste exercício - que consistia em verificar se a posição na matriz (através da fórmula ' $j * 10 + i$ ') correspondia ou não à posição pedida pelo usuário.

Para imprimir ou não as diferentes cartas pedidas pelo utilizador defini a seguinte estratégia – a variável ‘revelar’ é definida como ‘1’ caso a ‘posicaoAtual’ se encontre dentro das posições pedidas pelo usuário e assim imprimir na matriz as cartas pedidas, caso contrário, se ‘revelar’ for falso (0), será impresso a máscara ‘###’ no lugar da carta usando para esse efeito a constante ‘MASCARA’.

Em suma, o desafio que encontrei aqui foi o de encontrar a fórmula para calcular a posição na matriz, verificar se se encontrava dentro das posições pedidas pelo usuário e depois imprimir a carta correspondente ou a máscara.

Os diferentes testes realizados revelam o output de acordo com o esperado, ver anexos Alínea C.

No teste de qualidade referente a esta Alínea C podemos observar o mesmo problema da alínea anterior e também ‘EQ08: Indentação variável’ para as linhas 8 e 26 mas como são fornecidas pelo enunciado optei por não modificar.

#### **Alínea D:**

Nesta alínea continuo a recorrer a partes do código desenvolvido até aqui.

A estratégia que escolhi passou por começar a enumerar todas as cartas possíveis com a função ‘prepararBaralho’, para depois embaralhá-las no vetor ‘baralho’ através da função ‘baralhar’ já antes vista.

Para iniciar com a matriz 8 x 8 com ‘###’ (mascara) em todas as posições, optei por elaborar a função ‘exibirMatrizMascarada’.

Em seguida, através da função ‘realizarJogada’ o usuário é convidado a introduzir as posições que pretende ver na matriz e estas são recebidas por meio da função ‘receberJogada’; depois, recorrendo à função ‘validarJogada’ verifico se estas jogadas respeitam as regras enunciadas no exercício e em seguida para divulgar o resultado na matriz utilizo a função ‘exibirMatriz’.

No que diz respeito à regra de retirar as cartas do baralho com o mesmo operador, ou o mesmo número ou mesmo naipe concebi a função 'compararCartas' para indicar à função 'exibirMatriz' se a condição de igualdade é verdadeira. Caso as posições no vetor baralho resultem em '-1', são eliminadas.

Caso o usuário coloque nenhuma opção válida, é retornado 'JOGADA\_INVALIDA' e o jogo entra no modo automático, caso apenas uma opção válida seja inserida é retornado 'JOGADA\_INSUFICIENTE' e a jogada é pulada. Por fim, caso alguma opção já removida do baralho seja inserida, é retornado 'JOGADA\_REPETIDA' e o jogo termina.

Utilizo a função 'mostrarResultado' para revelar, caso existam, as cartas restantes na matriz final. Ao final é mostrado a quantidade de jogadas e a quantidade de cartas restantes.

Em resumo, utilizo as funções referidas até aqui na função principal deste código para atingir o objetivo final deste jogo, ou seja, utilizei diversas funções com o objetivo de gerar os números pseudoaleatórios, embaralhar as cartas, receber o pedido do usuário e exibi-las ou não na matriz de acordo com as regras pedidas no exercício.

Os diferentes testes realizados revelam o output de acordo com o esperado, ver anexos Alínea D.

O teste de qualidade referente a este exercício foi realizado, ver anexo, mas por falta de tempo não foi corrigido os erros apontados.

## Anexos

### Alínea A:

#### Casos testes pedidos:

Entrada	Saída
-1	Carta invalida
100	Carta invalida
0	+1A
51	/1D
21	-2B
19	-1D
41	*3B

#### Validação pre-check no VPL

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char operacoes[] = {'+', '-', '*', '/'};
6     char numeros[] = {'1', '2', '3', '4'};
7     char letras[] = {'A', 'B', 'C', 'D'};
8
9     // Criar 64 codigos possiveis
10    for (int i = 0; i < 4; i++) {
11        for (int j = 0; j < 4; j++) {
12            for (int k = 0; k < 4; k++) {
13                // ...
14            }
15        }
16    }
17
18    // Pedir para usuario colocar numero, guarda-lo e validar se numero inteiro +
19    int vCodigo;
20    if (scanf("%d", &vCodigo) != 1 || vCodigo < 0 || vCodigo >= 64) {
21        printf("carta invalida\n");
22        return 1;
23    }
24
25    // Imprimir a vCodigo correspondente ao código inserido
26    int i = vCodigo / 16;
27    int j = (vCodigo % 16) / 4;
28    int k = vCodigo % 4;
29    printf("%c%c%c\n", operacoes[i], numeros[j], letras[k]);
30    return 0;
31 }
```

Run

Pre-check

#### Evaluation:

##### -Summary of tests

```
>+-----+
>| 7 tests run/ 7 tests passed |
>+-----+
```

#### Casos testes extras:

Entrada	Saída
63	/4D
64	Carta invalida
&	Carta invalida
M	Carta invalida

#### Teste qualidade:

```
C:\Users\cmfmc\OneDrive\Bureau\qualidade>qualidade2 2203524A.c
#EQ4: Nomes sem significado
18|    int vCodigo;
```

## Alínea B:

### Casos testes pedidos:

Entrada	Saída
0	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] *3B *1C +2B *2A /1D /4D -1C /3D [2] +3B /2B -2B *4A *2D *4D /4C +2D [3] +4A *4B +3D *3D -4D +2A -1B /3A [4] -4B +4B /2D +2C -2D /2A +1A -3A [5] *1B /1B -3D /2C -1D /3B *4C -2C [6] *1D /4B /1A +4C *2C -4A *3A +4D [7] /3C -3B /4A *2B +3A +1C /1C *3C [8] +3C -1A +1B -3C -2A +1D -4C *1A </pre>
1	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] *1D /3A /2C *4C -1B *2A *2C *4A [2] +1A /1B /1C /2A *3C -2A -4D *4B [3] -4A -2B -3C -3A +2D /3D /3C *3A [4] -1C -3B +4C /4B +2B -1D /3B *3B [5] +3A *3D /4A /1D +4A /1A -3D +2C [6] *2B -4B /2B -2D /4C +1B *2D +1D [7] -1A /2D +3B -4C +3D +4D *4D *1A [8] +2A +3C +1C +4B *1B /4D -2C *1C </pre>

### Validação pre-check no VPL

```

1 #include <stdio.h>
2
3 #define NPM_CARTAS 64
4 #define CARTAS_POR_LINHA 8
5
6 unsigned int randaux() {
7     static long seed = 1;
8     return ((seed = seed * 214013L + 2531011L) >> 16) & 0xffff;
9 }
10
11 void Baralhar(int v[], int n) {
12     int i, j, aux;
13     // processar todos os elementos
14     for (i = 0; i < n - 1; i++) {
15         // gerar um valor aleatório para sortear o elemento do vector a ficar na posição i (entre i e n-1)
16         j = randaux() % (n - i);
17         aux = v[i];
18         v[i] = v[j];
19         v[j] = aux;
20     }
21 }
22
23 void main() {
24     int desperdício, i;
25     scanf("%d", &desperdício);
26
27     for(i=0; i<desperdício; i++)
28         randaux();
29
30     char operacoes[] = {"+", "-", "*", "/", "%"};
31     char numeros[] = {"1", "2", "3", "4"};
32     char letras[] = {"A", "B", "C", "D"};
33
34     int orden[NPM_CARTAS];
35
36     // Preencher a orden original
37     for (int i = 0; i < NPM_CARTAS; i++) {
38         orden[i] = i;
39     }
40
41     // Embaralhar a orden utilizando a função Baralhar
42     Baralhar(orden, NPM_CARTAS);
43
44     // Imprimir as 64 cartas possíveis por ordem de coluna
45     printf("  [1] [2] [3] [4] [5] [6] [7] [8]\n");
46     for (int i = 0; i < CARTAS_POR_LINHA; i++) {
47         printf("%5s", i < 10 ? "" : " ");
48         for (int j = 0; j < NPM_CARTAS / CARTAS_POR_LINHA; j++) {
49             int numeroJogador = orden[i + j * CARTAS_POR_LINHA];
50             int k = numeroJogador / 16;
51             int l = (numeroJogador % 16) / 4;
52             int m = numeroJogador % 4;
53             printf(" %5s%5s", operacoes[k], numeros[l]);
54             if (m < 3) printf(" ");
55         }
56         printf("\n");
57     }
58 }

```

Run ☒ Pre-check ☒

Evaluation:

Summary of tests

> 2 tests run/ 2 tests passed |

### Casos testes extra:

Entrada	Saída	
2	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] /4C /3B *3D -2B -4C *3C +4C /2C [2] *3B /1B /1A /4A -1D *1D *2C -4B [3] +4B -1B -3B +2D /1D +3D -4A +2A [4] /1C *2B -1A /2B -3C *1A *4A +3C [5] -2C +1B +4D *2D *4D *1B *1C +2C [6] *3A *4B +1C /2D +3B /4B -3D -4D [7] /3A /3D +1A -3A /4D +3A *4C +4A [8] -1C +1D -2D /2A -2A *2A /3C +2B </pre>	Colocar outro número 2 pex → os números são diferentes
0 10	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] *3B *1C +2B *2A /1D /4D -1C /3D [2] +3B /2B -2B *4A *2D *4D /4C +2D [3] +4A *4B +3D *3D -4D +2A -1B /3A [4] -4B +4B /2D +2C -2D /2A +1A -3A [5] *1B /1B -3D /2C -1D /3B *4C -2C [6] *1D /4B /1A +4C *2C -4A *3A +4D [7] /3C -3B /4A *2B +3A +1C /1C *3C [8] +3C -1A +1B -3C -2A +1D -4C *1A </pre>	Coloquei o 10 e o código se compila o 0, o primeiro número inteiro

Entrada	Saída	
0	Códigos iniciais função Baralhar: 41 9 12  [1] [2] [3] [4] [5] [6] [7] [8] [1] *3B *1C +2B *2A /1D /4D -1C /3D [2] +3B /2B -2B *4A *2D *4D /4C +2D [3] +4A *4B +3D *3D -4D +2A -1B /3A [4] -4B +4B /2D +2C -2D /2A +1A -3A [5] *1B /1B -3D /2C -1D /3B *4C -2C [6] *1D /4B /1A +4C *2C -4A *3A +4D [7] /3C -3B /4A *2B +3A +1C /1C *3C [8] +3C -1A +1B -3C -2A +1D -4C *1A	Coloquei como no enunciado para ver os codigos iniciais funcao baralhar
1	Códigos iniciais função Baralhar: 35 0 28  [1] [2] [3] [4] [5] [6] [7] [8] [1] *1D /3A /2C *4C -1B *2A *2C *4A [2] +1A /1B /1C /2A *3C -2A -4D *4B [3] -4A -2B -3C -3A +2D /3D /3C *3A [4] -1C -3B +4C /4B +2B -1D /3B *3B [5] +3A *3D /4A /1D +4A /1A -3D +2C [6] *2B -4B /2B -2D /4C +1B *2D +1D [7] -1A /2D +3B -4C +3D +4D *4D *1A [8] +2A +3C +1C +4B *1B /4D -2C *1C	Coloquei como no enunciado para ver os codigos iniciais funcao baralhar

### Teste qualidade

```
C:\Users\cmfmc\OneDrive\Bureau\qualidade>qualidade2 2203524B.c
#EQ4: Nomes sem significado
12|    int i, j, aux;
```



Alínea C:

Casos testes pedidos:

Entrada	Saída
0 55 71 16 33	<pre>[1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## -1C ### [2] ### ## ## ## ## ## ## [3] ### ## +3D ### ## ## ## [4] ### ## ## ## ## ## ## [5] ### ## ## ## -1D ### ## [6] *1D ### ## ## ## ## ## [7] ### ## ## ## ## ## ## [8] ### ## ## ## ## ## ##</pre>
1 18 45 91 53	<pre>[1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## ## ## ## ## ## [2] ### ## ## ## ## ## ## [3] ### ## ## ## ## ## ## [4] ### ## ## ## ## ## ## [5] ### ## ## ## /1D ### ## [6] ### ## ## ## ## ## ## [7] ### ## ## ## ## ## ## [8] +2A ### ## ## ## ## ##</pre>

Validação pre-check no VPL

```
1 #include <stdio.h>
2
3 #define NML_CARTAS 64
4 #define CARTAS_POR_LINHA 8
5 #define MASCARA "###"
6
7 unsigned int randaux() {
8     static long seed = 1;
9     return (((seed = seed * 214013L + 2531011L) >> 16) & 0x0fff);
10 }
11
12 void Baralhar(int v[], int n) {
13     int i, j, aux;
14     // processar todos os elementos
15     for (i = 0; i < n - 1; i++) {
16         j = i + randaux() % (n - i);
17         aux = v[i];
18         v[i] = v[j];
19         v[j] = aux;
20     }
21 }
22
23 void main() {
24     int desperdicio, i;
25     printf("Máscara: %s", MASCARA);
26     for (i = 0; i < desperdicio; i++)
27         randaux();
28
29     char operacoes[] = {"+", "-", "*", "/", ""};
30     char numeros[] = {"1", "2", "3", "4"};
31     char letras[] = {"A", "B", "C", "D"};
32
33     int orden[NML_CARTAS];
34
35     // Preencher a ordem original
36     for (int i = 0; i < NML_CARTAS; i++) {
37         orden[i] = i;
38     }
39
40     // Embaralhar a ordem utilizando a função Baralhar
41     Baralhar(orden, NML_CARTAS);
42
43     // Pedir ao usuário para digitar 4 números inteiros correspondentes a posições na matriz
44     int posicoes[4];
45     for (int i = 0; i < 4; i++) {
46         scanf("%d", &posicoes[i]);
47     }
48
49     int trava = 0;
50     for (int i = 0; i < 4; i++) {
51         if (trava == 0 && (posicoes[i] < 11 || posicoes[i] > 80)) {
52             trava = 1;
53         }
54         if (trava == 1) {
55             posicoes[i] = 0;
56         }
57     }
58
59     // Imprimir as 64 cartas possíveis por ordem de coluna
60     printf(" [1] [2] [3] [4] [5] [6] [7] [8]\n");
61     for (int i = 1; i <= CARTAS_POR_LINHA; i++) {
62         printf("%s", MASCARA);
63         for (int j = 1; j <= NML_CARTAS / CARTAS_POR_LINHA; j++) {
64             int numeroJogador = orden[(i - 1) * CARTAS_POR_LINHA + (j - 1)];
65             int k = numeroJogador / 16;
66             int l = (numeroJogador % 16) / 4;
67             int m = numeroJogador % 4;
68
69             // Calcular a posição atual na matriz
70             int posicaoAtual = j * 16 + i;
71
72             // Verificar se a posição atual está nas posições especificadas pelo usuário
73             int revelar = 0;
74             for (int n = 0; n < 4; n++) {
75                 if (posicaoAtual == posicoes[n]) {
76                     revelar = 1;
77                     break;
78                 }
79             }
80
81             if (revelar) {
82                 printf("%s", MASCARA);
83                 printf("%s", operacoes[k], numeros[l], letras[m]);
84             } else {
85                 printf("%s", MASCARA);
86             }
87         }
88         printf("\n");
89     }
90 }
```

Run Pre-check

Evaluation:

-Summary of tests

> 2 tests run/ 2 tests passed

Casos testes extra:

Entrada	Saída
& 55 71 16 33	<pre>[1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## ## ## ## ## ## [2] ### ## ## ## ## ## ## [3] ### ## ## ## ## ## ## [4] ### ## ## ## ## ## ## [5] ### ## ## ## ## ## ## [6] ### ## ## ## ## ## ## [7] ### ## ## ## ## ## ## [8] ### ## ## ## ## ## ##</pre>

Se por erro for digitado & não e revelado nenhuma posicao

Entrada	Saída	
0 10 71 16 33	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## ## ## ## ## ## ## [2] ### ## ## ## ## ## ## ## [3] ### ## ## ## ## ## ## ## [4] ### ## ## ## ## ## ## ## [5] ### ## ## ## ## ## ## ## [6] ### ## ## ## ## ## ## ## [7] ### ## ## ## ## ## ## ## [8] ### ## ## ## ## ## ## ## </pre>	A posição 10 sai fora da matriz 8 x 8 não sendo processado mais nada
0 55711633	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## ## ## ## ## ## ## [2] ### ## ## ## ## ## ## ## [3] ### ## ## ## ## ## ## ## [4] ### ## ## ## ## ## ## ## [5] ### ## ## ## ## ## ## ## [6] ### ## ## ## ## ## ## ## [7] ### ## ## ## ## ## ## ## [8] ### ## ## ## ## ## ## ## </pre>	Sequência de números inteiros na mesma linha, não são separados por espaços
0 55 71 16 33 44	<pre> [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ## ## ## ## ## -1C ## [2] ### ## ## ## ## ## ## ## [3] ### ## +3D ## ## ## ## ## [4] ### ## ## ## ## ## ## ## [5] ### ## ## ## -1D ## ## ## [6] *1D ## ## ## ## ## ## ## [7] ### ## ## ## ## ## ## ## [8] ### ## ## ## ## ## ## ## </pre>	O quinto número introduzido na sequência, 44, como só são pedidos 4 números por isso o 5 número não é revelado na matriz.

Teste qualidade:

```

C:\Users\cmfmc\OneDrive\Bureau\qualidade>qualidade2 2203524C.c
#EQ4: Nomes sem significado
13|    int i, j, aux;
#EQ8: Indentação -> variável (valor: 1)
8|    static long seed = 1;
26|    scanf("%d", &desperdicio);

```

## Alínea D:

### Casos testes pedidos:

Entrada	Saída	Entrada	Saída
0	<div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] -1C [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### +3D [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### -1D [8] ###</div> <div>[6] *1D [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] *3B [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] +3B [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### +4B [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### /1B [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] /3D [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### +3D [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] *3C [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] /4D [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### /2D [5] ### [6] ### /2A [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### /1A [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>Jogo com 6 jogadas. Restam 49 cartas.</div> </div>	1	<div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>[1] [2] [3] [4] [5] [6] [7] [8]</div> <div>[1] ### [2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[2] ### [3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[3] ### [4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[4] ### [5] ### [6] ### [7] ### [8] ###</div> <div>[5] ### [6] ### [7] ### [8] ###</div> <div>[6] ### [7] ### [8] ###</div> <div>[7] ### [8] ###</div> <div>[8] ###</div> <div>Jogo com 77 jogadas. Restam 0 cartas.</div> </div>
18 45 91 53		22 77 63	
41 78 38		85 76 56	
64 73 68		84 74 24	
53 66 13		61 48 14	
58 34 12		11 33 21	
65 55 87		32 82 83	
26 16 15		46 81 52	
35 18 75		51 25 37	
28 57 88		31 43 27	
54 36 86		23 17 71	
67 47 45		42 44 62	
72		0	



Casos testes extra:

Entrada	Saída	
0 12 15 11 12 24 -1 81 87 33 11	<pre>[1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### ### [2] ### ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### ### [4] ### ### ### ### ### ### ### ### [5] ### ### ### ### ### ### ### ### [6] ### ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ### ###  [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### ### [2] +3B ### ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### ### [4] ### ### ### ### ### ### ### ### [5] *1B ### ### ### ### ### ### ### ### [6] ### ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ### ###  [1] [2] [3] [4] [5] [6] [7] [8] [1] *3B ### ### ### ### ### ### ### ### [2] +3B ### ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### ### [4] ### +4B ### ### ### ### ### ### ### ### [5] ### ### ### ### ### ### ### ### [6] ### ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ### ###  [1] [2] [3] [4] [5] [6] [7] [8] [1]   ### ### ### ### ### ### ### ### [2]   ### ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### ### [4] ###   ### ### ### ### ### ### ### ### [5] ### ### ### ### ### ### ### ### [6] ### ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ### ### Jogo com 3 jogadas. Restam 61 cartas.</pre>	Permite que o usuário digite somente dois números.  Neste teste a posição -1 não existe, logo não é mostrada.  Jogo termina com 4 jogadas pois já tinha sido eliminado a posição 11.

Entrada	Saída	
-1	<pre>...Program finished with exit code 0 Press ENTER to exit console.</pre>	O desperdício deve ser numero inteiro positivo
Entrada	Saída	
0 24 35 91 1 12 15 81 87 35 15	<pre>       [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### [2] ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### [4] ### ### ### ### ### ### ### [5] ### ### ### ### ### ### ### [6] ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ###       [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### [2] ### ### ### ### ### ### ### [3] ### ### ### ### ### ### ### [4] ### +4B ### ### ### ### ### [5] ### ### -3D ### ### ### ### [6] ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ###       [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### [2] +3B ### ### ### ### ### ### [3] ### ### ### ### ### ### ### [4] ### ### ### ### ### ### ### [5] *1B ### ### ### ### ### ### [6] ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ###       [1] [2] [3] [4] [5] [6] [7] [8] [1] ### ### ### ### ### ### ### [2]   ### ### ### ### ### ### [3] ### ### ### ### ### ### ### [4] ### ### ### ### ### ### ### [5]   ### ### ### ### ### ### [6] ### ### ### ### ### ### ### [7] ### ### ### ### ### ### ### [8] ### ### ### ### ### ### ### Jogo com 3 jogadas. Restam 62 cartas.</pre>	<p>Aqui temos 91 e 1 fora da matriz.</p> <p>Só duas jogadas e continua o jogo.</p> <p>Jogo termina com 3 jogadas pois já tinha sido eliminado a posição 15.</p>

## Validação pre-check no VPL

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4
5 #define NUM_CARTAS 64
6 #define CARTAS_POR_LINHA 8
7 #define MASCARA "###"
8
9 #define JOGADA_INSUFICIENTE -3
10 #define JOGADA_INVALIDA -2
11 #define JOGADA_REPETIDA -1
12 #define JOGADA_VALIDA 1
13
14 // Gera um número pseudoaleatório a partir de uma sequência de sementes.
15 unsigned int randaux();
16
17 // Obtém um valor de desperdício do usuário e gera uma semente.
18 bool gerarSemente();
19
20 // Preenche o vetor e o embaralha baseado na semente atual.
21 void prepararBaralho(int baralho[], int n);
22
23 // Embaralha o vetor de cartas.
24 void embaralhar(int baralho[], int n);
25
26 // Exibe a matriz inicial com máscara.
27 void exibirMatrizMascarada();
28
29 // Limpa o buffer de entrada.
30 void limparBuffer();
31
32 // Recebe a jogada do usuário.
33 int receberJogada(int jogadas[]);
34
35 // Verifica se pelo menos uma jogada está entre 11 e 88, e mascara as que não estiverem.
36 int validarJogada(int baralho[], int jogadas[], int n);
37
38 int realizarJogada(int baralho[], int jogadas[], int n, bool eliminar);
39
40 int realizarJogadaAutomatica(int baralho[]);
```

Run  Pre-check 

**Evaluation:**

**Summary of tests**

```
>+-----+
>| 2 tests run/ 2 tests passed |
>+-----+
```

## Teste qualidade:

Por falta de tempo, não foram corrigidos no código, peço desculpa.

```
C:\Users\cmfmc\OneDrive\Bureau\qualidade>qualidade2 2203524D.c
#EQ3: Variáveis com nomes quase iguais (valor: 11)
235|     int validade_jogada = validarJogada(baralho, jogadas, num_jogadas);
234|     int num_jogadas = receberJogada(jogadas);
#EQ4: Nomes sem significado
67|     int quantidade_cartas_removidas = 0;
#EQ8: Indentação não variável (valor: 4)
52|     int baralho[NUM_CARTAS];
390|         printf(" %s", MASCARA);
#EQ23: Ciclos contendo apenas uma variável lógica como teste de paragem
69|     while(true) {
#EQ26: Nomes muito longos (valor: 27)
67|     int quantidade_cartas_removidas = 0;
```