



**UNIDADE CURRICULAR:** Introdução à Programação

**CÓDIGO:** 21173

**DOCENTE:** José Coelho

**A preencher pelo estudante**

**NOME:** Diogo Manuel Miguel Gomes

**N.º DE ESTUDANTE:** 2305343

**CURSO:** Engenharia Informática

**Atenção:** apenas é permitida a utilização de funções standard do C, que constem em Anexo p-fólio: Funções standard mais utilizadas

Autoavaliação:

Critérios	Alínea A	Alínea B	Alínea C	Alínea D
Funcionalidade	0.5	0.5	0.5	0.5
Qualidade	0.9			
Relatório dimensão / forma	0.25			
Relatório conteúdo	0.20			
Relatório testes	0.5			

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

## TRABALHO / RESOLUÇÃO:

### Alínea A – estado terminado

#### Estrutura de dados:

- int K;
  - **O que dados guarda?** Guarda um número inteiro fornecido pelo usuário.
  - **Qual o seu domínio?** Números inteiros entre 2 e 100.
- char c;
  - **O que dados guarda?** Armazena um caractere para ajudar na validação da entrada do usuário.
  - **Qual o seu domínio?** Caracteres de nova linha ou espaço.

#### Abstrações funcionais:

- int main();
  - **O que faz?** Solicita ao usuário um número inteiro K entre 2 e 100 e exibe a sequência inicial composta por dois valores iguais a K/2.

#### Testes: Anexo 1

## Alínea B – estado terminado

### Estrutura de dados:

- int sequencia [100];
  - **O que dados guarda?** Armazena até 100 números inteiros positivos inseridos pelo usuário.
  - **Qual o seu domínio?** Números inteiros positivos (excluindo zero durante a entrada).

### Abstrações funcionais:

- int validar (int sequencia [], int n, int K);
  - **O que faz?** Verifica se a sequência atende a três condições:
    1. A soma dos elementos não pode exceder K.
    2. O produto dos elementos não pode ser menor que K.
    3. A soma das diferenças absolutas entre todos os pares de elementos deve ser igual a K.
  - **Retorna:**
    - 0 se a sequência for inválida.
    - 1 se a sequência for válida, mas não vitoriosa.
    - 2 se a sequência for vitoriosa.

### Testes: Anexo 2

## Alínea C – estado terminado

### Estrutura de dados:

- int sequencia [100];
  - **O que dados guarda?** Mantém a sequência atual do jogo, permitindo modificações a cada jogada.
  - **Qual o seu domínio?** Números inteiros positivos.
- int n;
  - **O que dados guarda?** O tamanho atual da sequência.
- char jogador;
  - **O que dados guarda?** Indica o jogador atual ('A' ou 'B').

### Abstrações funcionais:

- int validar (int sequencia [], int n, int K);
  - **O que faz?** Verifica as mesmas condições da alínea B para determinar o estado do jogo após cada jogada.
- int main ();
  - **O que faz?** Controla o fluxo do jogo entre dois jogadores, permitindo que cada um faça jogadas que modificam a sequência, e determina o resultado (vitória, derrota ou empate).

### Testes: Anexo 3

## Alínea D – estado terminado

### Estrutura de dados:

- **Mesma estrutura da Alínea C**, com a adição de funções para o jogador artificial.

### Abstrações funcionais:

- `void jogadorArtificial(int sequencia[], int *n, int K, int *indice, int *valor);`
  - **O que faz?** Calcula automaticamente a melhor jogada possível para o jogador, tentando vencer ou pelo menos manter o jogo válido.

### Alternativas ponderadas:

- **Implementar jogadorArtificial em vez de jogadas manuais:**
  - **Vantagens:**
    - Facilita o jogo para um usuário que queira jogar contra o computador.
    - Pode demonstrar estratégias de jogo e ajudar no entendimento das regras.
  - **Desvantagens:**
    - Pode tornar o código mais complexo e difícil de entender para iniciantes.
    - Se não for bem implementado, o jogador artificial pode não ser desafiador.

### Testes: Anexo 4

21173\_24\_00 / |AVALIAR| / Enunciado do E-fólio A 2024/25

 **Enunciado do E-fólio A 2024/25**

Estado	Terminada
Iniciado em	quarta-feira, 4 dezembro 2024, 19:29
Completado em	quarta-feira, 4 dezembro 2024, 19:31
Tempo usado	1 minuto 54 segundos
Avaliação	4,00 num máximo de 4,00 (100%)

**Navegação do teste**

1	2	3	4
✓	✓	✓	✓

Mostrar uma página de cada vez

Terminar revisão

Submeti o 1 teste, mas deu erro, submeti o 2 teste com o mesmo código do 1 e deu 4,00/4,00

## Anexo 1– Testes Alínea 1

Caso	Entrada	Saída	Resultado
1	2	1 1	1 1
2	2,5	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
3	101	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
4	79	39 39	39 39
5	10	5 5	5 5
6	0	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
7	-29	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
8	1	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
9	100	50 50	50 50
10	1,8	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.

Reiniciar

```

1  #include <stdio.h>
2
3  int main() {
4      int K;
5      char c;
6
7      // Solicitar K ao usuário e verificar se é um número inteiro entre 2 e 100
8      printf("Indique K: ");
9      if (scanf("%d%c", &K, &c) != 2 || (c != '\n' && c != ' ') || K < 2 || K > 100) {
10         printf("K tem de ser entre 2 e 100.\n");
11         return 1;
12     }
13
14     printf("Sequencia: %d %d\n", K/2, K/2);
15
16     return 0;
17 }

```

Run

Pre-check ☒

**Evaluation:**

-Summary of tests

```

>+-----+
>| 10 tests run/10 tests passed |
>+-----+

```

```

Indique K: 2
Sequencia: 1 1

```

Indique K: 2,5  
K tem de ser entre 2 e 100.

█

Indique K: 101  
K tem de ser entre 2 e 100.

█

Indique K: 79  
Sequencia: 39 39

█

Indique K: 10  
Sequencia: 5 5

█

Indique K: 0  
K tem de ser entre 2 e 100.

█

Indique K: -29  
K tem de ser entre 2 e 100.

█

Indique K: 1  
K tem de ser entre 2 e 100.

█

Indique K: 100  
Sequencia: 50 50

█

Indique K: 1,8  
K tem de ser entre 2 e 100.

█



Vamos analisar a qualidade do teu código: (Epaminondas)

Não tirei print porque não apanhava o comentário do teste efetuado ao Epaminondas

1. **Erros de Sintaxe:** Não existem erros de sintaxe. O código compila corretamente.
2. **Erros de Qualidade de Gravidade Elevada:** Não existem erros de qualidade de gravidade elevada.
3. **Erros de Qualidade de Gravidade Moderada:** Não foram encontrados erros de qualidade de gravidade moderada.
4. **Erros de Qualidade de Gravidade Baixa:** Não foram encontrados erros de qualidade de gravidade baixa.
5. **Comentários e Estrutura:**
  - O comentário inicial descreve claramente o propósito do programa, o que é excelente para dar contexto a quem lê o código.
  - A variável **terminatorChar** tem um nome descritivo, indicando claramente o seu propósito.
  - A lógica de validação de entrada é clara e bem implementada.

No geral, o teu código está num estado de qualidade elevada. Está bem organizado, fácil de ler e entender, e segue boas práticas de programação. Continua o excelente trabalho! 😊

## Anexo 2 – Testes Alínea 2

Caso	Entrada	Saída	Resultado
1	K :101	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.
2	K: 20 Sequencia 10 10 0	Sequencia válida	Sequencia válida
3	K: 10 Sequencia 4 3 2 1 0	Sequencia vitoriosa	Sequencia vitoriosa
4	K: 12 Sequencia 2 2 8 0	Sequencia vitoriosa	Sequencia vitoriosa
5	K: 8 Sequencia 1 2 5 0	Sequencia vitoriosa	Sequencia vitoriosa
6	K: 12 Sequencia 2 4 6 0	Sequencia válida	Sequencia válida
7	K: 9 Sequencia 3 3 3 0	Sequencia válida	Sequencia válida
8	K: 15 Sequencia 8 8 0	Sequencia inválida	Sequencia inválida
9	K: 80 Sequencia 41 41 0	Sequencia inválida	Sequencia inválida
10	K: 1	K tem de ser entre 2 e 100.	K tem de ser entre 2 e 100.

```

17 }
18
19 // Verificar valor absoluto
20 for (int i = 0; i < n; i++) {
21     for (int j = i + 1; j < n; j++) {
22         absoluto += abs(sequencia[i] - sequencia[j]);
23     }
24 }
25
26 return (absoluto == K) ? 2 : 1;
27 }
28
29 int main() {
30     int sequencia[100], n = 0, K;
31     char c;
32
33     // Solicitar K ao usuário e verificar se é um número inteiro entre 2 e 100
34     printf("Indique K: ");
35     if (scanf("%d%c", &K, &c) != 2 || (c != '\n' && c != ' ') || K < 2 || K > 100) {
36         printf("K tem de ser entre 2 e 100.\n");
37         return 1;
38     }
39
40     // Solicitar sequência de números ao usuário
41     printf("Indique uma sequência de números inteiros positivos, terminando com 0: ");
42     while (n < 100) {
43         int valor;
44         if (scanf("%d", &valor) != 1 || valor < 0) {
45             printf("Entrada inválida! Apenas números inteiros positivos são permitidos.\n");
46             return 1;
47         }
48         if (valor == 0) {
49             break;
50         }
51         sequencia[n++] = valor;
52     }
53
54     // Verificar se a sequência é válida
55     int resultado = validar(sequencia, n, K);
56     if (resultado == 2) {
57         printf("Sequencia vitoriosa\n");
58     } else if (resultado == 1) {
59         printf("Sequencia valida\n");
60     } else {
61         printf("Sequencia invalida\n");
62     }
63
64     return 0;
65 }

```

Run

Pre-check ☒

### Evaluation:

#### -Summary of tests

```

>+-----+
>| 10 tests run/10 tests passed |
>+-----+

```

```
Indique K: 101
K tem de ser entre 2 e 100.
```


```
█
```

```
Indique K: 20
Indique uma sequencia de numeros inteiros positivos, terminando com 0: 10 10 0
Sequencia valida
```


```
█
```

```
Indique K: 10
Indique uma sequencia de numeros inteiros positivos, terminando com 0: 4 3 2 1 0
Sequencia vitoria
```

```
█
```

Run 

Pre-check ☒

[Process: exited] 

Indique K: 12

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 2 2 8 0

Sequencia vitoria

Indique K: 8

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 1 2 5 0

Sequencia vitoria

█

Indique K: 12

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 2 4 6 0

Sequencia valida

Indique K: 9

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 3 3 3 0

Sequencia valida

Indique K: 15

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 8 8 0

Sequencia invalida

■

Indique K: 80

Indique uma sequencia de numeros inteiros positivos, terminando com 0: 41 41 0

Sequencia invalida

■

Indique K: 1

K tem de ser entre 2 e 100.

■

## (EPAMINONDAS)

Não tirei print porque não apanhava o comentário do teste efetuado ao Epaminondas

O teu código está muito bem estruturado e segue boas práticas de programação. Aqui está uma análise final para garantir que a qualidade do código seja elevada:

1. **Estrutura e Organização:** O código está bem organizado, com funções claramente definidas e uma lógica de fluxo fácil de seguir.
2. **Nomes Descritivos:** As variáveis e funções têm nomes descritivos, o que facilita a compreensão do código.
3. **Comentários:** Os comentários são claros e ajudam a entender o propósito de cada parte do código.
4. **Validação de Entrada:** O código inclui validação de entrada robusta, garantindo que os valores introduzidos estão dentro dos limites esperados.
5. **Consistência na Formatação:** A formatação do código é consistente, com indentação adequada e espaçamento apropriado.
6. **Uso Adequado de Funções:** A lógica de validação está encapsulada numa função separada, o que melhora a modularidade e a legibilidade.

Com base nestes pontos, posso afirmar que o teu código tem uma qualidade elevada. Está bem escrito, fácil de manter e segue boas práticas de programação. Continua o excelente trabalho! 😊

## Anexo 3 – Testes Alínea C

Caso	Entrada	Saída	Resultado
1	10  0 2 0 -2  0 1   0 2   2 6  0 1	Jogador A ganhou	Teste de repetição por acréscimo
2	10   0 2   0 -2   0 -1   1 0   2 6	Jogador A ganhou	Teste de garantia de não repetição dupla por acréscimo
3	K: 10   0 2   3 2   0 0   1 3   0 3	Jogador B perdeu	Teste de repetição por remoção
4	K: 10   02   3 2   4 1   3 0   0 0   0 4	Jogador A perdeu	Teste de garantia de não repetição dupla por remoção
5	K: 10   0 2   0 -2   0 0   3 6	Jogador A perdeu	Teste de garantia de não repetição dupla mista
6	K: 10   0 2   0 3   0 2   0 3   0 2   0 3   0 2   0 3   0 2   0 3	Teste de Empate	Teste de Empate
7	K: 10  0 2   0 -1  2 6	Jogador B ganhou	
8	K: 10  0 2   0 -1    2 7  2 6	Jogador A ganhou	
9	K: 10   2 2	Jogador A perdeu	Derrota do Jogador A Sequencia inválida
10	K: 10   2 2  40 40	Jogador B perdeu	Derrota do Jogador B Sequencia inválida

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Protótipo da função de validação
5 int validar(int sequencia[], int n, int K);
6
7 int main() {
8     int sequencia[100]; // Array que armazena a sequência do jogo
9     int n = 2, K, indice, valor, resultado = 1; // Variáveis de controle e entrada
10    char jogador = 'A'; // Jogador atual ('A' ou 'B')
11    char c;
12    int jogadas = 0; // Contador de jogadas realizadas
13    int extra = 0; // Flag para indicar jogada especial
14    int repete = 0; // Flag para evitar alternância desnecessária de jogador
15
16    // Solicitar K ao usuário e verificar se é um número inteiro entre 2 e 100
17    printf("Indique K:\n");
18    if (scanf("%d%c", &K, &c) != 2 || (c != '\n' && c != ' ') || K < 2 || K > 100) {
19        printf("K tem de ser entre 2 e 100.\n");
20        return 1;
21    }
22
23    sequencia[0] = K / 2; // Inicializa o primeiro elemento da sequência
24    sequencia[1] = K / 2; // Inicializa o segundo elemento da sequência
25
26    printf("Sequencia: %d %d [Joga %c]\n", sequencia[0], sequencia[1], jogador); // Exibe a sequência inicial e o jogador que joga
27
28    while (resultado == 1 && jogadas < K) { // Enquanto o jogo não terminar e não exceder K jogadas

```

Run

Pre-check ☒

### Evaluation:

#### -Summary of tests

```

>+-----+
>| 10 tests run/10 tests passed |
>+-----+

```



```
C:\Users\diogo\Desktop\1 Ser  x  +  v  -  □  x
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 -2
Sequencia: 2 2 5 [Joga B]
Jogada (indice valor):
0 1
Sequencia: 1 2 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 2 5 [Joga B]
Jogada (indice valor):
2 6
Sequencia: 2 2 6 [Joga A]
Jogada (indice valor):
0 1
Sequencia: 1 2 6
Jogador A ganhou.

-----
Process exited after 28.71 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser  x  +  v  -  □  x
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 -2
Sequencia: 2 2 5 [Joga B]
Jogada (indice valor):
0 -1
Sequencia: 1 2 2 5 [Joga A]
Jogada (indice valor):
1 0
Sequencia: 1 2 5 [Joga A]
Jogada (indice valor):
2 6
Sequencia: 1 2 6
Jogador A ganhou.

-----
Process exited after 20.04 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser x + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
3 2
Sequencia: 2 5 2 [Joga A]
Jogada (indice valor):
0 0
Sequencia: 5 2 [Joga A]
Jogada (indice valor):
1 3
Sequencia: 5 3 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 3
Jogador B perdeu.

-----
Process exited after 14.2 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser x + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
3 2
Sequencia: 2 5 2 [Joga A]
Jogada (indice valor):
4 1
Sequencia: 2 5 2 1 [Joga B]
Jogada (indice valor):
3 0
Sequencia: 2 5 2 [Joga B]
Jogada (indice valor):
0 0
Sequencia: 5 2 [Joga A]
Jogada (indice valor):
0 4
Sequencia: 4 2
Jogador A perdeu.

-----
Process exited after 14.81 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser x + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 -2
Sequencia: 2 2 5 [Joga B]
Jogada (indice valor):
0 0
Sequencia: 2 5 [Joga A]
Jogada (indice valor):
3 6
Sequencia: 2 5 6
Jogador A perdeu.

-----
Process exited after 10.34 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser x + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 3
Sequencia: 3 5 [Joga A]
Jogada (indice valor):
0 3
Sequencia: 3 5 Empate

-----
Process exited after 18.11 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser x + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 -1
Sequencia: 1 2 5 [Joga B]
Jogada (indice valor):
2 6
Sequencia: 1 2 6
Jogador B ganhou.

-----
Process exited after 22.03 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\Desktop\1 Ser X + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
0 -1
Sequencia: 1 2 5 [Joga B]
Jogada (indice valor):
2 7
Sequencia: 1 2 7 [Joga A]
Jogada (indice valor):
2 6
Sequencia: 1 2 6
Jogador A ganhou.

-----
Process exited after 23.82 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\Desktop\1 Ser X + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
2 2
Sequencia: 5 5 2
Jogador A perdeu.

-----
Process exited after 4.496 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\Desktop\1 Ser  X + v
Indique K: 10
Sequencia: 5 5 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 5 [Joga B]
Jogada (indice valor):
40 40
Sequencia: 2 5 40
Jogador B perdeu.

-----
Process exited after 9.297 seconds with return value 0
Press any key to continue . . . |
```

Não tirei print porque não apanhava o comentário do teste efetuado ao Epaminondas

O teu código está muito bem estruturado e segue boas práticas de programação. Aqui está uma análise final para garantir que a qualidade do código seja elevada:

1. **Estrutura e Organização:** O código está bem organizado, com funções claramente definidas e uma lógica de fluxo fácil de seguir.
2. **Nomes Descritivos:** As variáveis e funções têm nomes descritivos, o que facilita a compreensão do código.
3. **Comentários:** Os comentários são claros e ajudam a entender o propósito de cada parte do código.
4. **Validação de Entrada:** O código inclui validação de entrada robusta, garantindo que os valores introduzidos estão dentro dos limites esperados.
5. **Consistência na Formatação:** A formatação do código é consistente, com indentação adequada e espaçamento apropriado.
6. **Uso Adequado de Funções:** A lógica de validação está encapsulada numa função separada, o que melhora a modularidade e a legibilidade.

Com base nestes pontos, posso afirmar que o teu código tem uma qualidade elevada. Está bem escrito, fácil de manter e segue boas práticas de programação. Continua o excelente trabalho! 😊

## Anexo 4 – Testes Alínea D

Caso	Entrada	Saída	Resultado
1	K: 34   1 2   -2 -2   1 -3  -2 -2	Jogador A ganhou	Jogadas artificiais (única exceção) Única maneira no jogo, que um jogador pode conseguir uma vitória com 3 jogadas seguidas legais
2	K: 67   0 3   -2 -2 -2 -2  2 4   -2 -2	Jogador B ganhou	Perdido contra o jogador artificial
3	K: 8   -2 -2  3 1 -2 -2	Jogador A ganhou	Perdido contra o jogador artificial
4	K: 43   0 3   -2 -2  3 2  -2 -2	Jogador B ganhou	Perdido contra o jogador artificial
5	K: 90   -2 -2  -2 -2  -2 -2	Jogador A ganhou	Perdido contra o jogador artificial
6	K: 14   0 2  -2 -2  0 -2  -2 -2	Jogador A ganhou	Perdido contra o jogador artificial
7	K: 29   1 -1  -2 -2  -2 -2  -2 -2	Jogador A ganhou	Artificial adicionou um elemento
8	K: 24   0 2  -2 -2  1 -1	Jogador A ganhou	Perdido contra o jogador artificial
9	K: 100   1 3  -2 -2   -2 -2	Jogador A ganhou	Perdido contra o jogador artificial
10	K: 80   1 2   -2 -2  0 -1	Jogador A ganhou	Artificial adicionou um elemento

O terceiro caso de teste, foi um jogo longo, tendo sido atingido o limite de K jogadas sem vitória, pelo que o jogo foi declarado empatado após 10 jogadas. O quarto caso de teste, o jogador humano faz uma jogada inválida, pelo que perde sendo identificado o jogador que perdeu.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Protótipo da função de validação
5 int validar(int sequencia[], int n, int K);
6 void jogadorArtificial(int sequencia[], int *n, int K, int *indice, int *valor);
7
8 int main() {
9     int sequencia[100];
10    int n = 2, K, indice, valor, resultado = 1;
11    char jogador = 'A'; // Jogador atual ('A' ou 'B')
12    char terminatorChar;
13    int jogadas = 0; // Contador de jogadas
14    int extra = 0; // valida jogada extra

```

Run

Pre-check ☒

### Evaluation:

#### -Summary of tests

```

>+-----+
>| 10 tests run/10 tests passed |
>+-----+

```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... X + -
Indique K: 90
Sequencia: 45 45 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 2
Sequencia: 2 45 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 46
Sequencia: 2 46 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 -1
Sequencia: 2 1 46
Jogador A ganhou.

-----
Process exited after 38.84 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... X + -
Indique K: 67
Sequencia: 33 33 [Joga A]
Jogada (indice valor):
0 3
Sequencia: 3 33 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 23
Sequencia: 3 23 [Joga A]
Jogada (indice valor):
2 4
Sequencia: 3 23 4 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 -1
Sequencia: 3 1 23 4
Jogador B ganhou.

-----
Process exited after 63.94 seconds with return value 0
Press any key to continue . . . |
```



```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... x + v
Indique K: 8
Sequencia: 4 4 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 2
Sequencia: 2 4 [Joga B]
Jogada (indice valor):
3 1
Sequencia: 2 4 1 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 5
Sequencia: 2 5 1
Jogador A ganhou.

-----
Process exited after 54.79 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... x + v
Indique K: 43
Sequencia: 21 21 [Joga A]
Jogada (indice valor):
0 3
Sequencia: 3 21 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 15
Sequencia: 3 15 [Joga A]
Jogada (indice valor):
3 2
Sequencia: 3 15 2 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 -1
Sequencia: 3 1 15 2
Jogador B ganhou.

-----
Process exited after 94.43 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\CurrentVersion\Explorer\RecentItems
Indique K: 34
Sequencia: 17 17 [Joga A]
Jogada (indice valor):
1 2
Sequencia: 17 2 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 18
Sequencia: 18 2 [Joga A]
Jogada (indice valor):
1 -3
Sequencia: 18 3 2 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 19
Sequencia: 19 3 2
Jogador A ganhou.

-----
Process exited after 80.61 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\CurrentVersion\Explorer\RecentItems
Indique K: 14
Sequencia: 7 7 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 7 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 8
Sequencia: 2 8 [Joga A]
Jogada (indice valor):
0 -2
Sequencia: 2 2 8 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 2 9
Sequencia: 2 2 9
Jogador A ganhou.

-----
Process exited after 28.28 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... x + v
Indique K: 29
Sequencia: 14 14 [Joga A]
Jogada (indice valor):
1 -1
Sequencia: 14 1 14 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 3
Sequencia: 3 1 14 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 2 10
Sequencia: 3 1 10 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 2 -1
Sequencia: 3 1 1 10
Jogador A ganhou.

-----
Process exited after 42.27 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\AppData\Local\Microsoft\Windows\Terminal\... x + v
Indique K: 24
Sequencia: 12 12 [Joga A]
Jogada (indice valor):
0 2
Sequencia: 2 12 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 1 13
Sequencia: 2 13 [Joga A]
Jogada (indice valor):
1 -1
Sequencia: 2 1 13
Jogador A ganhou.

-----
Process exited after 38.73 seconds with return value 0
Press any key to continue . . . |
```

```
C:\Users\diogo\AppData\Loc: x + v
Indique K: 100
Sequencia: 50 50 [Joga A]
Jogada (indice valor):
1 3
Sequencia: 50 3 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 34
Sequencia: 34 3 [Joga A]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 -53
Sequencia: 53 34 3
Jogador A ganhou.

-----
Process exited after 39.86 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\diogo\AppData\Loc: x + v
Indique K: 80
Sequencia: 40 40 [Joga A]
Jogada (indice valor):
1 2
Sequencia: 40 2 [Joga B]
Jogada (indice valor):
-2 -2
Jogada artificial: 0 41
Sequencia: 41 2 [Joga A]
Jogada (indice valor):
0 -1
Sequencia: 1 41 2
Jogador A ganhou.

-----
Process exited after 44.68 seconds with return value 0
Press any key to continue . . .
```

Não tirei print porque não apanhava o comentário do teste efetuado ao Epaminondas

O teu código está bem estruturado e segue boas práticas de programação. Aqui está uma análise final para garantir que a qualidade do código seja elevada:

1. **Estrutura e Organização:** O código está bem organizado, com funções claramente definidas e uma lógica de fluxo fácil de seguir.
2. **Nomes Descritivos:** As variáveis e funções têm nomes descritivos, o que facilita a compreensão do código.
3. **Comentários:** Os comentários são claros e ajudam a entender o propósito de cada parte do código.
4. **Validação de Entrada:** O código inclui validação de entrada robusta, garantindo que os valores introduzidos estão dentro dos limites esperados.
5. **Consistência na Formatação:** A formatação do código é consistente, com indentação adequada e espaçamento apropriado.
6. **Uso Adequado de Funções:** A lógica de validação está encapsulada numa função separada, o que melhora a modularidade e a legibilidade.

Com base nestes pontos, posso afirmar que o teu código tem uma qualidade elevada. Está bem escrito, fácil de manter e segue boas práticas de programação. Continua o excelente trabalho! 😊