EDB1 - Prof. Eiji Adachi

Atividade Avaliativa Diagnóstica em C++

Objetivo: Avaliar o entendimento dos alunos sobre estruturas de controle, repetição, funções e manipulação de arrays em C++. A atividade será realizada em formato de exercícios práticos.

Instruções Gerais:

- **Implementação**: Cada aluno deve implementar a função auxiliar solicitada em cada exercício. Os nomes das funções e variáveis devem estar em **português**.
- **Testes na `main`**: A função `main` implementa alguns testes executáveis; se todos os testes passarem, há grande chance, mas não garantia, de que sua implementação está correta.
- **Nomenclatura de Arquivos**: Cada exercício deve ser salvo em um arquivo separado com o nome `qN.cpp`, onde `N` é o número da questão (por exemplo, `q1.cpp`, `q2.cpp`, etc.).
- **Código Base**: Utilize exatamente o código base fornecido para cada questão, adicionando apenas a implementação solicitada. Digite todo o código aqui do roteiro, não faça copiar e colar.

Exercício 1: Número é Par ou Ímpar

- **Arquivo**: `q1.cpp`
- **Descrição**: Escreva uma função que determine se um número é par. A função deve retornar `true` se o número for par e `false` caso contrário.

```cpp
#include <iostream>
using namespace std;

bool ehPar(int numero) {
    // Implementar a lógica para verificar se o número é par
}

int main() {
    int testesPassados = 0;

    cout << "Teste 1: O resultado esperado é 1 e o valor retornado foi "
<< ehPar(4) << endl;
    if (ehPar(4) == true) testesPassados++;

    cout << "Teste 2: O resultado esperado é 0 e o valor retornado foi "
<< ehPar(7) << endl;
    if (ehPar(7) == false) testesPassados++;

    cout << "Teste 3: O resultado esperado é 1 e o valor retornado foi "
<< ehPar(0) << endl;
    if (ehPar(0) == true) testesPassados++;

    cout << "Teste 4: O resultado esperado é 0 e o valor retornado foi "
<< ehPar(-5) << endl;
```

```
    if (ehPar(-5) == false) testesPassados++;

    cout << "Teste 5: O resultado esperado é 1 e o valor retornado foi "
<< ehPar(-8) << endl;
    if (ehPar(-8) == true) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 2: Número é Primo

- **Arquivo**: `q2.cpp`
- **Descrição**: Crie uma função que verifique se um número é primo. A função deve retornar `true` se o número for primo e `false` caso contrário.

```
#include <iostream>
using namespace std;

bool ehPrimo(int numero) {
    // Implementar a lógica para verificar se o número é primo
}

int main() {
    int testesPassados = 0;

    cout << "Teste 1: O resultado esperado é 1 e o valor retornado foi "
<< ehPrimo(5) << endl;
    if (ehPrimo(5) == true) testesPassados++;

    cout << "Teste 2: O resultado esperado é 0 e o valor retornado foi "
<< ehPrimo(10) << endl;
    if (ehPrimo(10) == false) testesPassados++;

    cout << "Teste 3: O resultado esperado é 1 e o valor retornado foi "
<< ehPrimo(13) << endl;
    if (ehPrimo(13) == true) testesPassados++;

    cout << "Teste 4: O resultado esperado é 0 e o valor retornado foi "
<< ehPrimo(1) << endl;
    if (ehPrimo(1) == false) testesPassados++;

    cout << "Teste 5: O resultado esperado é 1 e o valor retornado foi "
<< ehPrimo(17) << endl;
    if (ehPrimo(17) == true) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;
```

```
    return 0;
}
```

Exercício 3: Fatorial Iterativo

- **Arquivo**: `q3.cpp`
- **Descrição**: Implemente uma função que calcule o fatorial de um número inteiro positivo usando um método iterativo.

```
#include <iostream>
using namespace std;

int fatorial(int numero) {
    // Implementar a lógica para calcular o fatorial
}

int main() {
    int testesPassados = 0;

    cout << "Teste 1: O resultado esperado é 120 e o valor retornado foi "
<< fatorial(5) << endl;
    if (fatorial(5) == 120) testesPassados++;

    cout << "Teste 2: O resultado esperado é 1 e o valor retornado foi "
<< fatorial(0) << endl;
    if (fatorial(0) == 1) testesPassados++;

    cout << "Teste 3: O resultado esperado é 1 e o valor retornado foi "
<< fatorial(1) << endl;
    if (fatorial(1) == 1) testesPassados++;

    cout << "Teste 4: O resultado esperado é 6 e o valor retornado foi "
<< fatorial(3) << endl;
    if (fatorial(3) == 6) testesPassados++;

    cout << "Teste 5: O resultado esperado é 3628800 e o valor retornado
foi " << fatorial(10) << endl;
    if (fatorial(10) == 3628800) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 4: Divisíveis por 3 e 5

- **Arquivo**: `q4.cpp`
- **Descrição**: Escreva uma função que conte quantos números entre 0 e N são divisíveis por 3 e 5 ao mesmo tempo.

```cpp
#include <iostream>
using namespace std;

int contarDivisiveisPor3e5(int N) {
    // Implementar a lógica para contar os números divisíveis por 3 e 5
}

int main() {
    int testesPassados = 0;

    cout << "Teste 1: O resultado esperado é 1 e o valor retornado foi "
<< contarDivisiveisPor3e5(15) << endl;
    if (contarDivisiveisPor3e5(15) == 1) testesPassados++;

    cout << "Teste 2: O resultado esperado é 2 e o valor retornado foi "
<< contarDivisiveisPor3e5(30) << endl;
    if (contarDivisiveisPor3e5(30) == 2) testesPassados++;

    cout << "Teste 3: O resultado esperado é 0 e o valor retornado foi "
<< contarDivisiveisPor3e5(10) << endl;
    if (contarDivisiveisPor3e5(10) == 0) testesPassados++;

    cout << "Teste 4: O resultado esperado é 3 e o valor retornado foi "
<< contarDivisiveisPor3e5(45) << endl;
    if (contarDivisiveisPor3e5(45) == 3) testesPassados++;

    cout << "Teste 5: O resultado esperado é 6 e o valor retornado foi "
<< contarDivisiveisPor3e5(90) << endl;
    if (contarDivisiveisPor3e5(90) == 6) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 5: Contar Números Primos até N

- **Arquivo**: `q5.cpp`
- **Descrição**: Crie uma função que conte quantos números entre 0 e N são primos.

```cpp
#include <iostream>
using namespace std;

int contarPrimosAteN(int N) {
```

```
    // Implementar a lógica para contar os números primos até N
}

int main() {
    int testesPassados = 0;

    cout << "Teste 1: O resultado esperado é 4 e o valor retornado foi "
<< contarPrimosAteN(10) << endl;
    if (contarPrimosAteN(10) == 4) testesPassados++;

    cout << "Teste 2: O resultado esperado é 8 e o valor retornado foi "
<< contarPrimosAteN(20) << endl;
    if (contarPrimosAteN(20) == 8) testesPassados++;

    cout << "Teste 3: O resultado esperado é 3 e o valor retornado foi "
<< contarPrimosAteN(5) << endl;
    if (contarPrimosAteN(5) == 3) testesPassados++;

    cout << "Teste 4: O resultado esperado é 0 e o valor retornado foi "
<< contarPrimosAteN(1) << endl;
    if (contarPrimosAteN(1) == 0) testesPassados++;

    cout << "Teste 5: O resultado esperado é 25 e o valor retornado foi "
<< contarPrimosAteN(100) << endl;
    if (contarPrimosAteN(100) == 25) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 6: Buscar Elemento no Array

- **Arquivo**: `q6.cpp`
- **Descrição**: Implemente uma função que busque um número em um array. A função deve retornar a posição do número no array ou `-1` se não estiver presente.

```
#include <iostream>
using namespace std;

int buscarElemento(int arr[], int tamanho, int alvo) {
    // Implementar a lógica de busca no array
}

int main() {
    int testesPassados = 0;

    int arr[] = {1, 2, 3, 4, 5};
```

```
    cout << "Teste 1: O resultado esperado é 2 e o valor retornado foi "
<< buscarElemento(arr, 5, 3) << endl;
    if (buscarElemento(arr, 5, 3) == 2) testesPassados++;

    cout << "Teste 2: O resultado esperado é -1 e o valor retornado foi "
<< buscarElemento(arr, 5, 6) << endl;
    if (buscarElemento(arr, 5, 6) == -1) testesPassados++;

    cout << "Teste 3: O resultado esperado é 0 e o valor retornado foi "
<< buscarElemento(arr, 5, 1) << endl;
    if (buscarElemento(arr, 5, 1) == 0) testesPassados++;

    cout << "Teste 4: O resultado esperado é 4 e o valor retornado foi "
<< buscarElemento(arr, 5, 5) << endl;
    if (buscarElemento(arr, 5, 5) == 4) testesPassados++;

    cout << "Teste 5: O resultado esperado é -1 e o valor retornado foi "
<< buscarElemento(arr, 5, -2) << endl;
    if (buscarElemento(arr, 5, -2) == -1) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 7: Encontrar Menor Elemento no Array

- **Arquivo**: `q7.cpp`
- **Descrição**: Crie uma função que encontre o menor elemento de um array de inteiros.

```
#include <iostream>
using namespace std;

int encontrarMenorElemento(int arr[], int tamanho) {
    // Implementar a lógica para encontrar o menor elemento
}

int main() {
    int testesPassados = 0;

    int arr1[] = {3, 1, 4, 1, 5};
    cout << "Teste 1: O resultado esperado é 1 e o valor retornado foi "
<< encontrarMenorElemento(arr1, 5) << endl;
    if (encontrarMenorElemento(arr1, 5) == 1) testesPassados++;

    int arr2[] = {10, 20, 5, 15};
    cout << "Teste 2: O resultado esperado é 5 e o valor retornado foi "
<< encontrarMenorElemento(arr2, 4) << endl;
    if (encontrarMenorElemento(arr2, 4) == 5) testesPassados++;
```

```
    int arr3[] = {-3, -1, -4, -2};
    cout << "Teste 3: O resultado esperado é -4 e o valor retornado foi "
<< encontrarMenorElemento(arr3, 4) << endl;
    if (encontrarMenorElemento(arr3, 4) == -4) testesPassados++;

    int arr4[] = {7};
    cout << "Teste 4: O resultado esperado é 7 e o valor retornado foi "
<< encontrarMenorElemento(arr4, 1) << endl;
    if (encontrarMenorElemento(arr4, 1) == 7) testesPassados++;

    int arr5[] = {0, 2, -1, 3};
    cout << "Teste 5: O resultado esperado é -1 e o valor retornado foi "
<< encontrarMenorElemento(arr5, 4) << endl;
    if (encontrarMenorElemento(arr5, 4) == -1) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 8: Somatório dos Elementos do Array

- **Arquivo**: `q8.cpp`
- **Descrição**: Escreva uma função que calcule o somatório dos elementos de um array de inteiros.

```
#include <iostream>
using namespace std;

int somatorioArray(int arr[], int tamanho) {
    // Implementar a lógica para calcular o somatório
}

int main() {
    int testesPassados = 0;

    int arr1[] = {1, 2, 3, 4};
    cout << "Teste 1: O resultado esperado é 10 e o valor retornado foi "
<< somatorioArray(arr1, 4) << endl;
    if (somatorioArray(arr1, 4) == 10) testesPassados++;

    int arr2[] = {5, 5, 5, 5};
    cout << "Teste 2: O resultado esperado é 20 e o valor retornado foi "
<< somatorioArray(arr2, 4) << endl;
    if (somatorioArray(arr2, 4) == 20) testesPassados++;

    int arr3[] = {0, 0, 0, 0};
    cout << "Teste 3: O resultado esperado é 0 e o valor retornado foi "
<< somatorioArray(arr3, 4) << endl;
```

```
    if (somatorioArray(arr3, 4) == 0) testesPassados++;

    int arr4[] = {-1, -2, -3, -4};
    cout << "Teste 4: O resultado esperado é -10 e o valor retornado foi "
<< somatorioArray(arr4, 4) << endl;
    if (somatorioArray(arr4, 4) == -10) testesPassados++;

    int arr5[] = {100, 200, 300};
    cout << "Teste 5: O resultado esperado é 600 e o valor retornado foi "
<< somatorioArray(arr5, 3) << endl;
    if (somatorioArray(arr5, 3) == 600) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 9: Calcular Média dos Elementos do Array

- **Arquivo**: `q9.cpp`
- **Descrição**: Implemente uma função que calcule a média dos elementos de um array de inteiros.

```
#include <iostream>
using namespace std;

double mediaArray(int arr[], int tamanho) {
    // Implementar a lógica para calcular a média
}

int main() {
    int testesPassados = 0;

    int arr1[] = {1, 2, 3, 4};
    cout << "Teste 1: O resultado esperado é 2.5 e o valor retornado foi "
<< mediaArray(arr1, 4) << endl;
    if (mediaArray(arr1, 4) == 2.5) testesPassados++;

    int arr2[] = {5, 5, 5, 5};
    cout << "Teste 2: O resultado esperado é 5 e o valor retornado foi "
<< mediaArray(arr2, 4) << endl;
    if (mediaArray(arr2, 4) == 5) testesPassados++;

    int arr3[] = {0, 0, 0, 0};
    cout << "Teste 3: O resultado esperado é 0 e o valor retornado foi "
<< mediaArray(arr3, 4) << endl;
    if (mediaArray(arr3, 4) == 0) testesPassados++;

    int arr4[] = {-1, -2, -3, -4};
    cout << "Teste 4: O resultado esperado é -2.5 e o valor retornado foi
```

```
" << mediaArray(arr4, 4) << endl;
    if (mediaArray(arr4, 4) == -2.5) testesPassados++;

    int arr5[] = {10, 20, 30, 40, 50};
    cout << "Teste 5: O resultado esperado é 30 e o valor retornado foi "
<< mediaArray(arr5, 5) << endl;
    if (mediaArray(arr5, 5) == 30) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Exercício 10: Elemento com Maior Frequência no Array

- **Arquivo**: `q10.cpp`
- **Descrição**: Escreva uma função que encontre o elemento que ocorre com maior frequência em um array de inteiros (valores entre 0 e 10). A função deve retornar o elemento que ocorre com maior frequência.

```
#include <iostream>
using namespace std;

int elementoMaisFrequente(int arr[], int tamanho) {
    // Implementar a lógica para encontrar o elemento mais frequente
}

int main() {
    int testesPassados = 0;

    int arr1[] = {3, 1, 2, 2, 3, 3}; // Não ordenado
    cout << "Teste 1: O resultado esperado é 3 e o valor retornado foi "
<< elementoMaisFrequente(arr1, 6) << endl;
    if (elementoMaisFrequente(arr1, 6) == 3) testesPassados++;

    int arr2[] = {5, 4, 4, 5, 5, 4}; // Não ordenado
    cout << "Teste 2: O resultado esperado é 4 ou 5 e o valor retornado
foi " << elementoMaisFrequente(arr2, 6) << endl;
    if (elementoMaisFrequente(arr2, 6) == 4 || elementoMaisFrequente(arr2,
6) == 5) testesPassados++;

    int arr3[] = {0, 1, 0, 0, 1, 1, 1}; // Não ordenado
    cout << "Teste 3: O resultado esperado é 1 e o valor retornado foi "
<< elementoMaisFrequente(arr3, 7) << endl;
    if (elementoMaisFrequente(arr3, 7) == 1) testesPassados++;

    int arr4[] = {1, 2, 3, 4, 5}; // Todos elementos únicos
    cout << "Teste 4: O resultado esperado é qualquer elemento (pois todos
têm a mesma frequência) e o valor retornado foi " <<
```

```
elementoMaisFrequente(arr4, 5) << endl;
    if (elementoMaisFrequente(arr4, 5) >= 1 && elementoMaisFrequente(arr4,
5) <= 5) testesPassados++;

    int arr5[] = {7, 8, 7, 8, 7, 8, 8}; // Não ordenado
    cout << "Teste 5: O resultado esperado é 8 e o valor retornado foi "
<< elementoMaisFrequente(arr5, 7) << endl;
    if (elementoMaisFrequente(arr5, 7) == 8) testesPassados++;

    cout << "Sua implementação passou em " << testesPassados << " de 5
testes." << endl;

    return 0;
}
```

Reflexão e Próximos Passos

- **Identifique as Questões com Dificuldades**: Anote quais questões você não conseguiu resolver e revisite-as.
- **Analise os Erros**:
    - **Erros de Lógica**: Revise a lógica da sua implementação. Verifique se você compreende o problema e se seguiu todos os requisitos.
    - **Erros de Sintaxe**: Certifique-se de que seu código está livre de erros de sintaxe e segue as convenções da linguagem.
    - **Casos Especiais**: Verifique se sua implementação trata corretamente casos especiais ou limites, como números negativos, zero ou arrays vazios.
- **Solicite Feedback**: Se possível, discuta suas dificuldades com colegas para obter diferentes perspectivas e soluções.
- **Pratique Mais**: Utilize recursos adicionais como livros, tutoriais online e exercícios práticos para fortalecer suas habilidades.