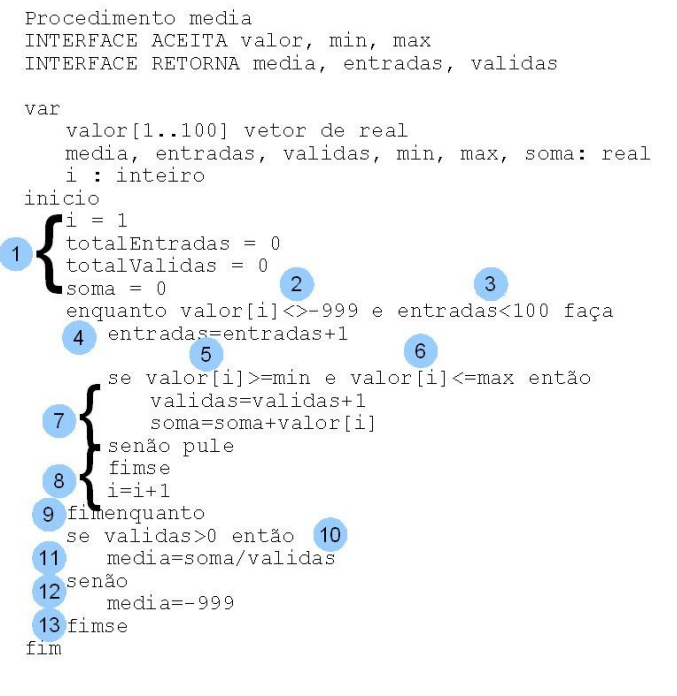
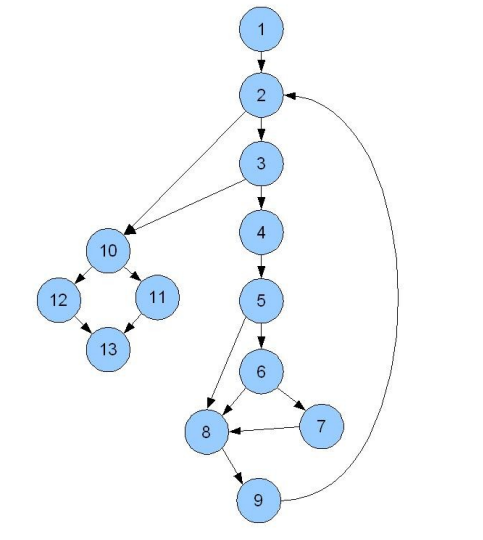
|  |  |  |
| --- | --- | --- |
| **Disciplina** | **Prof. Dacio Machado** | |
| PROJETO IMPLEMENTAÇÃO E TESTE DE SOFTWARE | Valor | +01 ATV |
| **ATIVIDADE: TESTE ESTRUTURAL** | Aluno: Andrey Eduardo Indras  RA: 23050861-2 | |
| **ESOFT - 6 - N** |

**Atividade prática de teste Estrutural Passos:**

1. Projetar **casos de teste estruturais** para avaliar os quatro algoritmos dos itens listados abaixo. Conforme o exemplo abaixo, e o excerto do Livro Didático.
2. Preencher os ARTEFATOS de teste abaixo para os testes projetados.
3. Construa, em sua linguagem de preferência os seguintes algoritmos:
   1. Um algoritmo que lê um número e imprime a lista dos seus divisores
   2. Um algoritmo que lê dois números e calcula o máximo divisor comum pelo método de Euclides.
   3. Um algoritmo que lê as 4 notas de um aluno e diga se ele passou por média, está em final ou reprovou
   4. Um algoritmo em que dado dois números *n* e *k* (n< k), calcule e apresente a combinatória de n elementos tomados k a k

**Exemplo de Desenvolvimento**: Derivar os casos de teste para um programa que calcula a média das entradas válidas, usando o método do caminho básico.



**Passo 1:** Desenhe o grafo de fluxo correspondente

**Passo 2:** Calcule a complexidade ciclomática. V(G) = 6 regiões V(G) = 17 arestas – 13 nós + 2 = 6 V(G) = 5 nós predicados + 1 = 6

**Passo 3:** Determine um conjunto base de caminhos independentes.

Caminho 1: 1-2-10-11-13

Caminho 2: 1-2-10-12-13

Caminho 3: 1-2-3-10-11-13

Caminho 4: 1-2-3-4-5-8-9-2...

Caminho 5: 1-2-3-4-5-6-8-9-2...

Caminho 6: 1-2-3-4-5-6-7-8-9-2…

**Passo 4:** Prepare os casos de teste que vão forçar a execução de cada caminho:

O caminho 1 só pode ser testado como parte dos caminhos 4, 5 e 6

Caminho 2: valor (i) = -999; resultados esperados: média = -999 e os outros valores com os valores iniciais.

Caminho 6: valor (i) = entrada válida; resultados esperados: média correta baseada em n valores e totais apropriados.

**PLANOS DE TESTE A SER DESCRITO:**

ITENS A TESTAR / ABORDAGEM:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N॰** | **Item** | **Especificação** |  | **ABORDAGEM:**  análise de fluxo de controle para números positivos, negativos e zero.  análise dos laços de repetição e condições de parada.  cobertura de todas as condições de aprovação/reprovação.  validação de entrada e cálculo fatorial. |
| 1 | Algoritmo de Divisores | Lê um número e imprime a lista dos seus divisores |
| 2 | Algoritmo MDC (Euclides) | Lê dois números e calcula o máximo divisor comum pelo método de Euclides |
| 3 | |  | | --- | |  |  |  | | --- | | Média de 4 notas | | Lê 4 notas de um aluno e determina se passou por média, está em final ou reprovou |
| 4 | Algoritmo de Combinatória | Dados dois números n e k (n<k), calcula a combinatória de n elementos tomados k a k |

CRONOGRAMA DE TESTES

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Tarefa** | **Início** | **Fim** | **Esforço** | **Pré** | **Pessoa** | **Obs** |
| 01 | Análise de código e criação de grafos de fluxo | 11/09/2025 | 11/09/2025 | **3h** |  | Andrey | Algoritmos: a, b, c, d |
| 02 | Cálculo da complexidade ciclomática | 11/09/2025 | 11/09/2025 | **1h** | 01 | Andrey | Para todos algoritmos |
| 03 | Definição de caminhos independentes | 11/09/2025 | 11/09/2025 | **2h** | 02 | Andrey | Casos de teste |
| 04 | Execução e validação dos testes | 11/09/2025 | 11/09/2025 | **3h** | 03 | Andrey | Relatório final |

AMBIENTE DE TESTE

|  |  |
| --- | --- |
| **Ambiente** | **Descrição** |
| Hardware | PC Intel i5, 8GB RAM, SSD 256GB |
| Software | Windows 10, Java JDK 21 |
| Ferramental | IDE IntelliJ, JUnit para execução dos testes, Lucidchart para grafos de fluxo |

IDENTIFICAÇÃO DE CASO DE TESTE / IDENTIFICAÇÃO DE PROCEDIMENTO DE TESTE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N॰** | **Caso de Teste** | **Identificação do Caso de Teste** |  | **Procedimento** | **Identificação do Procedimento de Teste** |
| 1 | Algoritmo de Divisores | CT01\_DIVISORES – CT03\_DIVISORES | Testar entradas válidas, inválidas e limite | PT01\_DIVISORES |
| 2 | Algoritmo MDC (Euclides) | CT03\_MDC – CT07\_MDC | Testar entradas iguais, múltiplos, primos, zero e inválidos | PT02\_MDC |
| 3 | Média de 4 notas | CT7\_MEDIA – CT11\_MEDIA | Testar cenários de aprovação, final, reprovação e inválido | PT03\_MEDIA |
| 4 | Algoritmo de Combinatória | CT11\_COMBINATORIA – CT13\_COMBINATORIA | Testar entradas válidas, inválidas e limite | PT04\_COMBINATORIA |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT01\_DIVISORES | |
| **Itens a Testar** | Algoritmo de divisores - entrada válida | |
| **Entradas** | **Campo** | **Valor** |
| Número | 12 |
|  |  |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 1, 2, 3, 4, 6, 12 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT01\_DIVISORES | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT02\_DIVISORES | |
| **Itens a Testar** | Algoritmo de divisores - caso limite | |
| **Entradas** | **Campo** | **Valor** |
| Número | 1 |
|  |  |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 1 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT01\_DIVISORES | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT03\_DIVISORES | |
| **Itens a Testar** | Algoritmo de divisores - entrada inválida | |
| **Entradas** | **Campo** | **Valor** |
| Número | -5 |
|  |  |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | "Número deve ser positivo" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT01\_DIVISORES | |
| **Dependência** | Nenhuma | |

PROCEDIMENTO DE TESTE

|  |  |
| --- | --- |
| **Identificação** | PT01\_DIVISORES |
| **Objetivo** | Validar cobertura de caminhos do algoritmo de divisores |
| **Requisitos** | Algoritmo implementado, ambiente de teste configurado |
| **Fluxo** | 1. Executar CT01 com número composto (12)  2. Verificar se todos os divisores são listados corretamente  3. Executar CT02 com caso limite (1)  4. Verificar retorno correto  5. Executar CT03 com entrada inválida (-5)  6. Verificar tratamento de erro  7. Documentar resultados |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT04\_MDC | |
| **Itens a Testar** | MDC - números iguais | |
| **Entradas** | **Campo** | **Valor** |
| A | 15 |
| B | 15 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 15 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT02\_MDC | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT05\_MDC | |
| **Itens a Testar** | MDC - números múltiplos | |
| **Entradas** | **Campo** | **Valor** |
| A | 48 |
| B | 18 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 6 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT02\_MDC | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT06\_MDC | |
| **Itens a Testar** | MDC - números primos entre si | |
| **Entradas** | **Campo** | **Valor** |
| A | 17 |
| B | 13 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 1 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT02\_MDC | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT07\_MDC | |
| **Itens a Testar** | MDC - entrada inválida | |
| **Entradas** | **Campo** | **Valor** |
| A | 0 |
| B | 5 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | "Números devem ser positivos e maiores que zero" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT02\_MDC | |
| **Dependência** | Nenhuma | |

PROCEDIMENTO DE TESTE

|  |  |
| --- | --- |
| **Identificação** | PT02\_MDC |
| **Objetivo** | Validar algoritmo de Euclides cobrindo todas as condições estruturais |
| **Requisitos** | Algoritmo MDC implementado, ambiente de teste configurado |
| **Fluxo** | 1. Executar CT04 com números iguais (15,15)  2. Verificar se retorna o próprio número  3. Executar CT05 com múltiplos (48,18)  4. Verificar cálculo correto através das iterações  5. Executar CT06 com primos entre si (17,13)  6. Verificar retorno 1  7. Executar CT07 com entrada inválida (0,5)  8. Verificar tratamento de erro  9. Documentar todos os caminhos percorridos |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT08\_MEDIA | |
| **Itens a Testar** | Média - aluno aprovado | |
| **Entradas** | **Campo** | **Valor** |
| Nota1 | 8.5 |
| Nota2 | 9.0 |
| Nota3 | 8.0 |
| Nota4 | 9.5 |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | Média = 8.75, Status: "APROVADO" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT03\_MEDIA | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT09\_MEDIA | |
| **Itens a Testar** | Média - aluno em final | |
| **Entradas** | **Campo** | **Valor** |
| Nota1 | 6.0 |
| Nota2 | 5.5 |
| Nota3 | 6.5 |
| Nota4 | 6.0 |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | Média = 6.0, Status: "FINAL" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT03\_MEDIA | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT10\_MEDIA | |
| **Itens a Testar** | Média - aluno reprovado | |
| **Entradas** | **Campo** | **Valor** |
| Nota1 | 3.0 |
| Nota2 | 4.0 |
| Nota3 | 3.5 |
| Nota4 | 4.5 |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | Média = 3.75, Status: "REPROVADO" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT03\_MEDIA | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT11\_MEDIA | |
| **Itens a Testar** | Média - entrada inválida | |
| **Entradas** | **Campo** | **Valor** |
| Nota1 | 8.0 |
| Nota2 | 7.0 |
| Nota3 | 6.0 |
| Nota4 | 11.0 |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | "Notas devem estar entre 0 e 10" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT03\_MEDIA | |
| **Dependência** | Nenhuma | |

PROCEDIMENTO DE TESTE

|  |  |
| --- | --- |
| **Identificação** | PT03\_MEDIA |
| **Objetivo** | Cobrir todas as condições de aprovação do sistema de notas |
| **Requisitos** | Sistema de média implementado, critérios definidos (>=7 aprovado, >=5 final, <5 reprovado) |
| **Fluxo** | 1. Executar CT08 com notas para aprovação (média >= 7.0)  2. Verificar cálculo e status "APROVADO"  3. Executar CT09 com notas para final (5.0 <= média < 7.0)  4. Verificar status "FINAL"  5. Executar CT10 com notas para reprovação (média < 5.0)  6. Verificar status "REPROVADO"  7. Executar CT11 com nota inválida (>10)  8. Verificar validação de entrada  9. Documentar cobertura de todos os ramos condicionais |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT12\_COMBINATORIA | |
| **Itens a Testar** | Combinatória - entrada válida | |
| **Entradas** | **Campo** | **Valor** |
| N | 5 |
| K | 3 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | 10 |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT04\_COMBINATORIA | |
| **Dependência** | Nenhuma | |

CASO DE TESTE

|  |  |  |
| --- | --- | --- |
| **Identificação** | CT13\_COMBINATORIA | |
| **Itens a Testar** | Combinatória - entrada inválida (n >= k) | |
| **Entradas** | **Campo** | **Valor** |
| N | 8 |
| K | 5 |
|  |  |
| **Saídas Esperadas** | **Campo** | **Valor** |
|  | "n deve ser menor que k" |
|  |  |
| **Ambiente** | Windows 10, Java JDK 21, IntelliJ | |
| **Procedimento** | PT04\_COMBINATORIA | |
| **Dependência** | Nenhuma | |

PROCEDIMENTO DE TESTE

|  |  |
| --- | --- |
| **Identificação** | PT04\_COMBINATORIA |
| **Objetivo** | Validar cálculo combinatorial e condição n < k |
| **Requisitos** | Algoritmo de combinatória implementado com validação n < k |
| **Fluxo** | 1. Executar CT12 com entrada válida (n=5, k=3)  2. Verificar cálculo: C(5,3) = 5!/(3!×2!) = 10  3. Executar CT13 com condição inválida (n>=k)  4. Verificar se a validação n < k funciona corretamente  5. Verificar tratamento de erro adequado  6. Documentar cobertura dos caminhos estruturais |