

Etapa 10 – Depuração e Tratamento de Erros (Debugging and Error Handling)

Trilha: Lógica de Programação

Introdução

- Mesmo programas simples podem conter **erros (bugs)**.
- A **depuração (debugging)** é o processo de **identificar e corrigir** esses erros.
- O **tratamento de erros (error handling)** garante que o programa reaja de forma **controlada** a falhas inesperadas.

“Erros são inevitáveis — a forma como o programa lida com eles define sua robustez.”

Tipos de Erros

1. Erro de Sintaxe (Syntax Error)

- Quebra das regras da linguagem.
- Exemplo: esquecer um parêntese ou ponto e vírgula.

2. Erro de Execução (Runtime Error)

- Ocorre durante a execução, como dividir por zero.

3. Erro Lógico (Logic Error)

- O programa executa, mas o resultado está errado.

Exemplo de Erros

- **Erro de Sintaxe:**

se $x > 10$

 escreva("Maior que 10")

fimfuncao

- Falta o “então” (then).

- **Erro de Execução:**

$a \leftarrow 10$

$b \leftarrow 0$

$c \leftarrow a / b$ // divisão por zero

- **Erro Lógico:**

$media \leftarrow soma / total + 1$ // lógica incorreta

O que é Depuração (Debugging)

- É o processo de **investigar e eliminar bugs**.
- Envolve **analisar variáveis, acompanhar a execução e testar hipóteses**.
- Ferramentas modernas permitem **executar passo a passo (step-by-step)**.
- Depurar é **pensar como o computador pensa**.

Técnicas de Depuração

- **Leitura de código (code review)**
- **Uso de mensagens de log (logging)**
- **Execução passo a passo (step over / step into)**
- **Pontos de interrupção (breakpoints)**
- **Análise de variáveis (watch / inspect)**

“O melhor depurador ainda é o programador que entende o que escreveu.”

Uso de Logs

- Inserir mensagens para acompanhar o fluxo do programa.
- Ajuda a entender onde o erro ocorre.
- **Exemplo:**

```
escreva("Iniciando cálculo")
resultado ← calcula_total()
escreva("Resultado:", resultado)
```

- **Boas práticas:**
 - Não exibir informações sensíveis.
 - Usar níveis de log (info, warning, error).
 - Remover logs excessivos antes da entrega final.

Tratamento de Erros (Error Handling)

- Permite que o programa **continue executando mesmo após um erro**.
- Impede que o sistema quebre completamente.
- **Estrutura genérica:**

tente

```
// código que pode gerar erro  
resultado ← dividir(a, b)
```

capture erro

```
    escreva("Ocorreu um erro: ", erro)
```

fimtente

- “Tratar erros é antecipar o inesperado.”

Erros Comuns em Programas

- Divisão por zero
- Variável não inicializada
- Índice fora do limite
- Falha de conversão de tipo
- Entrada inválida do usuário
- Sempre valide os dados **antes de processá-los.**

Estratégias de Tratamento

- Validação de entradas (input validation)
- Uso de valores padrão (default values)
- Mensagens claras para o usuário
- Uso de blocos try/catch (tente/capture)
- Retornos seguros (safe returns)
- Exemplo:

```
funcao dividir(a, b)
    se b = 0 entao
        escreva("Erro: divisão por zero.")
        retorna 0
    senao
        retorna a / b
    fimse
fimfuncao
```

Boas Práticas

- Antecipe possíveis falhas.
 - Use nomes de variáveis descritivos.
 - Evite mensagens genéricas como “Erro 404”.
 - Sempre teste seu programa em casos extremos.
- “Erros bem tratados fazem o usuário confiar no sistema.”

Ferramentas de Depuração

- Ambientes de desenvolvimento (IDEs) como:
 - Visual Studio Code
 - PyCharm
 - Eclipse
 - Dev-C++
- Ferramentas de log:
 - Logcat (Android)
 - Console (JavaScript / Python)
 - Log4j, Winston, etc.

Conclusão

- **Depuração** é o processo de entender e corrigir erros.
- **Tratamento de erros** é o cuidado de fazer o sistema reagir com controle.
- Bons desenvolvedores não evitam erros — **eles aprendem a lidar com eles.**
- **Resumo:**

“Depurar é compreender o erro; tratar é saber continuar sem quebrar.”

Conclusão

- **Busca e ordenação** são operações centrais em lógica e estruturas de dados.
- Conhecer suas diferenças e aplicações melhora a **eficiência** dos algoritmos.
- A escolha correta do método depende do **tamanho e organização** dos dados.
- **Resumo:**

“Buscar é encontrar; ordenar é preparar o caminho para encontrar mais rápido.”