

## AULA 10 - Instrumentação

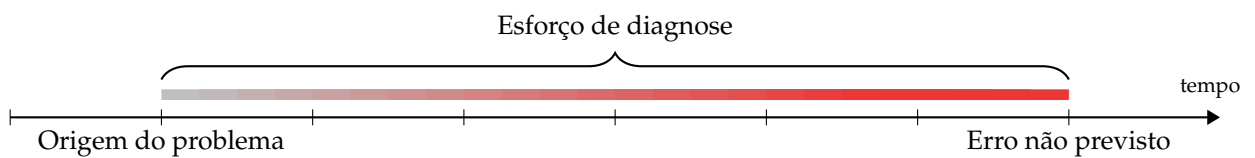
Iniciais dos alunos: *AVC, JPP, PC*

---

Disciplina: *Programação Modular (INF1301)* – Professor: *Flavio Bevilacqua*

:

### 1) Problema ao realizar testes



Colocamos controles ao longo do processo de desenvolvimento para minimizar o esforço de diagnose. Com controles, sabemos até qual ponto a aplicação estava correta e minimizamos possibilidades de erros.

Esforço de diagnose

- Grande
- Muito sujeito a erros

Contribui para esta dificuldade (agravantes)

- Não estabelece com facilidade a origem do problema.
- Tempo decorrido entre o instante da falha e o observado.
- Falhas intermitentes, acontecem de vez em quando.
- Causa externa ao código, falhas não relacionadas ao código da aplicação.

### 2) O que é Instrumentação?

Instrumentação são fragmentos de código de controle inseridos na aplicação durante o desenvolvimento de forma a monitorar a execução e minimizar o esforço de diagnose. Fragmentos de código de controle podem ser dados (redundantes)

ou comandos (blocos de controle). Eles não contribuem para o objetivo do programa, consomem recursos de execução e custam para serem desenvolvidos.

### 3) Objetivos

- Detectar falhas de funcionamento do programa o mais cedo possível de forma automática.
- Impedir que falhas se propaguem.
- Medir propriedades dinâmicas do programa.

### 4) Conceitos

- Programa robusto: intercepta a execução quando observa o problema e mantém o dano confinado.
- Programa tolerante a falhas: é robusto e possui mecanismo de recuperação, ou seja, o programa é preparado para condição hostil do ambiente (Ex: permite salvar progresso quando encontra erro).
- Deteriorização controlada: programa recupera de uma instabilidade e continua funcionando mesmo com uma perda de funcionalidade (subconjunto de tolerância a falhas).

### 5) Esquema de inclusão de instrumentos em C e C++:

Todo trecho de instrumentação fica entre `#ifdef _DEBUG` e `#endif`

```
#ifdef _DEBUG
```

```
    – Codigos
```

```
    – Dados
```

```
#endif
```

### 6) Assertivas Executáveis

Assertivas se tornam código para testá-las. Assertivas precisam ser corretas e

completas. Vantagens: informa um problema quase imediatamente após ter sido gerado, controle de integridade é feito pela máquina e reduz o risco de falha humana. Assertivas precisam ser corretas e completas.

#### 7) **Trace:**

Instrumento que apresenta um resultado ou mensagem quando é acionado em determinado ponto do programa.

Ex: printf

Existem dois tipos de trace, o de instrução e o de evolução. O trace de instrução executa no ponto onde é chamado. O trace de evolução executa quando uma variável ou estado é alterado(a).

A vantagem do trace é que a execução do programa é monitorada. A desvantagem é que seu uso pode gerar relatórios "poluídos".

#### 8) **Depurador:**

Instrumento que executa a aplicação passo a passo utilizando breakpoints para gerar resultados parciais.

#### 9) **Verificador de Estrutura de Dados:**

Instrumento que verifica a consistência de uma estrutura de dados implementando uma validação com base no modelo estrutural e nas assertivas estruturais.

10)