DCC024 Linguagens de Programação
2020/1

Tratamento de Erros

Haniel Barbosa

Tratamento de Erros

- Erros são comuns ao se escrever programas[*citation_needed*]
  1. Codificação errada
  2. Condições não tratadas durante codificação

Tratamento de Erros

- Erros são comuns ao se escrever programas[*citation_needed*]
  1. Codificação errada
  2. *Condições não tratadas durante codificação*

Tratamento de Erros

- Erros são comuns ao se escrever programas[*citation_needed*]
  1. Codificação errada
  2. *Condições não tratadas durante codificação*

- Como lidar com esses erros?

- Técnicas de tratamento de erros objetivam:
  - Prevenção de erros
  - Identificação de erros
  - Recuperação a partir de erros

Alguns tipos de tratamento de erros

O que pode dar de errado com o *pop* de uma pilha? Como tratar o erro?

Alguns tipos de tratamento de erros

O que pode dar de errado com o *pop* de uma pilha? Como tratar o erro?

- Guardas
- Definições totais
- Sinalização de erros (*error flagging*)
- Finalização da execução (*fatal error*)
- Pré e pós condições
- Exceções (*exceptions*)

O uso de exceções define um padrão de projeto

- Declaração de exceções
- Geração de exceções
- Captura e tratamento de exceções

Exceções devem ser usadas com cuidado

- Introduzem um fluxo de controle implícito
  - Exceções são essencialmente *goto* sofisticados
  - Código com exceções pode ser mais difícil de ler e compreender
- Bons usos de exceções

Exceções devem ser usadas com cuidado

- Introduzem um fluxo de controle implícito
  - Exceções são essencialmente *goto* sofisticados
  - Código com exceções pode ser mais difícil de ler e compreender
- Bons usos de exceções
  - Repassar o erro para o método com o contexto necessário para tratá-lo

Exceções devem ser usadas com cuidado

- Introduzem um fluxo de controle implícito
  - Exceções são essencialmente *goto* sofisticados
  - Código com exceções pode ser mais difícil de ler e compreender
- Bons usos de exceções
  - Repassar o erro para o método com o contexto necessário para tratá-lo
  - Bloquear o fluxo de controle no ponto do erro é melhor do que continuar após ele

Exceções devem ser usadas com cuidado

- Introduzem um fluxo de controle implícito
  - Exceções são essencialmente *goto* sofisticados
  - Código com exceções pode ser mais difícil de ler e compreender
- Bons usos de exceções
  - Repassar o erro para o método com o contexto necessário para tratá-lo
  - Bloquear o fluxo de controle no ponto do erro é melhor do que continuar após ele
- Maus usos: todos os outros :)

Em resumo...

- Exceções são eventos ”anormais” que podem ocorrer durante a execução de um programa, são caracterizados de maneira específica através de uma exceção, e permitem tratamento específico para tais situações.
- Mecanismos de tratamento de exceções em SML
  - Declaradas com *exception*, disparadas com *raise*, tratadas com *handle*
  - Possui exceções padrão (e.g., DIV, MATCH, ...)
- Mecanismos de tratamento de exceções em Python
  - Blocos *try*/*except*
  - Extensões de *Exception*
  - Possui exceções padrão (e.g., ZERODIVISIONERROR, VALUEERROR, ...)
  - Bloco *finally* permite especificar ações padrão que sempre são executadas
- Mecanismos de tratamento de exceções em C++
  - Qualquer tipo. Mas recomendado usar extensões de *std::exception*
  - Blocos *try*/*catch*
  - RAII anula necessidade de *finally* para limpeza de memória