Tratamento de excecões e tratamento de eventos

Marco A L Barbosa
malbarbo.pro.br

Departamento de Informática
Universidade Estadual de Maringá

Conteúdo

Introdução a tratamento de exceções

Introdução a tratamento de exceções

- A maioria dos computadores (hardware) são capazes de detectar certos erros em tempo de execução, como divisão por zero, falha de dispositivo de entrada e saída, etc
  - As primeiras linguagens de programação não forneciam nenhum mecanismo para os programas detectarem ou tratarem estes erros
  - Quando um erro acontecia, o programa era finalizado e o controle retornava para o sistema operacional que imprimia uma mensagem de erro na tela

Introdução a tratamento de exceções

- Outros erros podem ser detectados por software, como índice do arranjo fora do intervalo, final de arquivo, etc

Introdução a tratamento de exceções

- Em ambos os casos é interessante permitir que o programa reaja ao erro e possivelmente continue a sua execução

Introdução a tratamento de exceções

- Conceitos básicos
  - Uma **exceção** é um evento incomum, errôneo ou não, que é detectado por hardware ou software que precisa de processamento especial
  - O processamento especial que é necessário quando uma exceção é detectada é chamado de **tratamento de exceção**
  - Este processamento é feito por uma unidade de código chamada de **tratador de exceção**
  - Uma exceção é **lançada** (thrown, raised) quando seu evento associado ocorre

Introdução a tratamento de exceções

- Uma linguagem que não oferece mecanismos específicos de tratamento de exceção, não exclui a possibilidade de tratamento de exceções definidas pelo usuário e identificadas por software
  - Retorno de código de erro
  - Parâmetro auxiliar, que é usado como variável de estado
  - Rótulo, que é utilizado como endereço de retorno
  - Subprograma, que é utilizado como tratador da exceção

Introdução a tratamento de exceções

- Limitações
  - O código necessário para detectar erro é tedioso e polui o código
  - Se uma exceção não será tratada na unidade que ela ocorre, o tratador deve ser passado para todos os subprogramas na sequência de chamadas

Introdução a tratamento de exceções

- Vantagens de mecanismos de tratamento de exceção integrados a linguagem
  - Geração de código pelo compilador para checagem e geração de exceção
  - Propagação de exceção
  - Encorajamento do programador a considerar várias possíveis erros
  - Reuso de código de tratamento de exceção

Introdução a tratamento de exceções

- Questões de projeto
  - Como e onde os tratadores de exceções são especificados, quais são seus escopos?
  - Como a ocorrência de uma exceção é vinculada ao tratador de exceção?
  - Informações sobre as exceções podem ser passadas para o tratador?
  - Onde a execução continua, se é que continua, depois que o tratador da exceção completa a sua execução?
  - Alguma forma de finalização é fornecida?
  - Como as exceções definidas pelo usuário são especificadas?

Introdução a tratamento de exceções

- Questões de projeto
  - Se existem exceções pré-definidas, deve haver tratadores de exceção padrões para programas que não fornecem seus próprios tratadores?
  - As exceções pré-definidas podem ser explicitamente geradas?
  - Os erros detectáveis por hardware são tratados como exceções que podem ser manipuladas?
  - Existe alguma exceção pré-definida?
  - Deve ser possível desabilitar exceções pré-definidas?

Introdução a tratamento de exceções

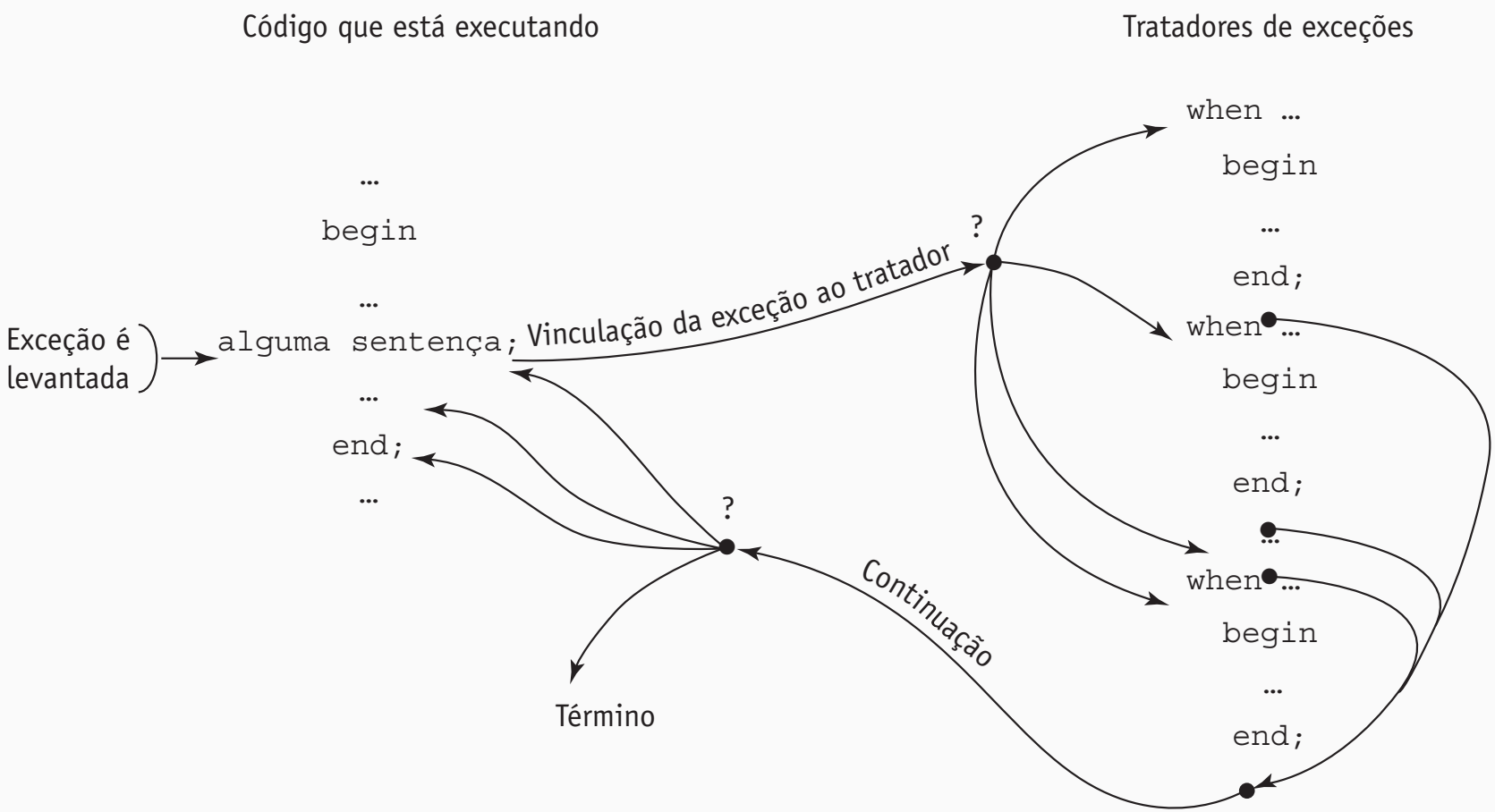

Figura 14.1 Fluxo de controle de tratamento de exceções.

Tratamento de exceções em Ada

Tratamento de exceções em Ada

- Baseado nas boas partes de PL/I e CLU
- Em geral os tratadores de exceções são locais ao código em que a exceção pode ocorrer, desta forma, não é necessário (nem possível) passar parâmetros para o tratador

Tratamento de exceções em Ada

- Forma do tratador

```
when exception_choice {| exception_choice} =>
  statement_sequence
```

- Forma do `exception_choice`

```
exception_name | others
```

Tratamento de exceções em Ada

- Tratadores podem ser colocados em blocos ou no corpo de subprogramas, pacotes ou tarefas
- Os tratadores são agrupados em uma cláusula `exception`, que deve aparecer no final do bloco ou unidade

```
begin
-- the block or unit body --
exception
  when exception_name_1 =>
    -- first handler --
  when exception_name_2 =>
    -- second handler --
    -- other handlers --
end;
```

Tratamento de exceções em Ada

- Vinculação das exceções aos tratadores
  - Quando um bloco ou unidade que gera uma exceção inclui um tratador para aquela exceção, a exceção é vinculada estaticamente ao tratador
  - Se o bloco ou unidade não inclui um tratador para aquela exceção, a exceção é propagada para ser manipulada em outro lugar
    - Procedimentos, propagada para o chamados
    - Bloco, propagada para o escopo em que ele aparece
    - Pacote corpo, propagada para a parte de declaração da unidade que declarou o pacote (se for uma unidade de biblioteca, o programa é terminado)
    - Tarefa, sem propagação, a tarefa é marcada como completada

Tratamento de exceções em Ada

- Vinculação das exceções aos tratadores
  - O bloco ou unidade que gera a exceção, juntamente com todos as unidades para os quais a exceção é propagada, mas que não manipularão a exceção, é sempre terminado
  - O controle sempre continua depois da cláusula de exceção, que é sempre o final do bloco ou unidade

Tratamento de exceções em Ada

- Outras decisões de projeto
  - Exceções definidas no pacote `Standard`: `Constraint_Error`, `Numeric_Error`, `Program_Error`, `Storage_Error`, `Tasking_Error`
  - Exceções definidas pelo usuário tem a forma: `exception_name_list: exception`
  - As exceções definidas pelo usuário são tratadas da mesma forma que as pré-definidas
  - Exceções são geradas com a cláusula `raise`: `raise [exception_name]`
  - Algumas exceções podem ser desativadas com a cláusula `pragma`: `pragma Suppress(check_name)`

Tratamento de exceções em Ada

- Exemplo (ruim) do livro
- O único uso adequado neste exemplo é a entrada de um número inválido
- Arquivo `grade_distribution.adb`

Tratamento de exceções em Ada

- Avaliação
  - Representa o consenso sobre exceções em 1980
  - Permite a propagação de exceção para escopos que não tem acesso a ela
  - Nem sempre é possível determinar a origem da exceção
  - Tratamento inadequado de exceções em tarefas
  - As exceções não foram adaptadas para trabalharem com objetos

Tratamento de exceções em C++

Tratamento de exceções em C++

- Adicionada a linguagem em 1990
- Baseada no projeto da CLU, Ada e ML

Tratamento de exceções em C++

- Os tratadores de exceções tem a forma

```
try {
  // code that is expected to raise a exception
}
catch (formal parameter) {
  // a handler body
}
...
catch (formal parameter) {
  // a handler body
}
```

Tratamento de exceções em C++

- Cada cláusula `catch` define um tratador
- Pode ter apenas um parâmetro formal, e o tipo tem que ser único
- Não é necessário definir o nome do parâmetro formal
- O parâmetro formal pode ser usado para transferir informações para o tratador
- O parâmetro formal pode ser `...`, o que defini um tratador para todas as exceções não manipuladas

Tratamento de exceções em C++

- Geração de exceções
  - `throw [expression]`

Tratamento de exceções em C++

- Vinculação das exceções aos tratadores
  - O tipo da expressão na cláusula `throw` seleciona o tratador
  - Um tratador com parâmetro formal do tipo `T`, `const T`, `T&`, `const T&`, ou qualquer tipo derivado de `T`, casa com uma expressão do tipo `T`
  - Quando uma exceção é gerada em uma cláusula `try`, a execução do código do `try` é interrompida
  - A busca por um tratador começa pelos tratadores que seguem o `try`

Tratamento de exceções em C++

- Vinculação das exceções aos tratadores
  - A busca por um tratador é sequencial
  - Se um tratador não é encontrado, a exceção é propagada
  - Quando a exceção chega a função principal, o tratador padrão é chamado
  - Quando a execução de um tratador `M` é completada, o fluxo de controle segue para a primeira instrução após o último tratador da sequencia de tratadores que `M` faz parte

Tratamento de exceções em C++

- Outras decisões de projeto
  - Todas as exceções são definidas pelo usuário
  - Existe um tratador padrão, chamado `unexpected`, que pode ser redefinido
  - As exceções podem ter qualquer tipo
  - Subprogramas podem listar as exceções que podem ser geradas na sua execução
  - Um subprograma que não liste as suas exceções, podem gerar qualquer exceção

Tratamento de exceções em C++

- Arquivo `grade_distribution.cpp`

Tratamento de exceções em C++

- Avaliação
  - Propagação de exceção semelhante ao Ada
  - Não existem exceções pré-definidas detectáveis por hardware
  - Problemas de legibilidade, qualquer tipo pode ser usado como exceção
  - É possível passar informações ao tratador

Tratamento de exceções em Java

Tratamento de exceções em Java

- Baseado em C++, alinhado com o POO
- Todas as exceções são objetos de classes que são descendentes de `Throwable`

Tratamento de exceções em Java

- `Throwable` tem duas subclasses
  - `Error`
    - A classe `Error` e suas descendentes são gerada pela JVM quando algum erro interno acontece, como falta de espaço no heap
    - Os programas de usuário não devem tratar estas exceções
  - `Exception`
    - A classe `Exception` e suas descendentes representam exceções que podem ser tratadas pelos programas dos usuários
    - Uma das subclasses de `Exception` é `RuntimeException`
    - Uma das subclasses de `RuntimeException` é `ArrayIndexOutOfBoundsException`

Tratamento de exceções em Java

- Tratadores de exceções
  - Semelhante ao do C++, exceto que cada `catch` requer um nome para o parâmetro, e o tipo do parâmetro precisa ser descendente de `Throwable`
  - As exceções são geradas com a cláusula `throw`

Tratamento de exceções em Java

- Vinculação das exceções aos tratadores
  - Semelhante ao C++
  - Uma exceção é vinculada ao primeiro tratador cujo parâmetro seja da mesma classe ou de uma classe ancestral

Tratamento de exceções em Java

- Outras decisões de projeto
- Exceções checadas e não checadas
  - Exceções das classes `Error` e `RuntimeException` e dos descendentes são chamadas de **exceções não checadas**
  - As outras exceções são chamadas de **exceções checadas**
  - As exceções checadas que podem ser geradas por um método, devem estar listadas na cláusula `throws` ou serem manipuladas no método

Tratamento de exceções em Java

- Um método não pode declarar mais exceções na cláusula `throws` do que o método que ele sobrescreve
- Um método que chama um método que lista um exceção checada em sua cláusula `throws`, tem três opções para lidar com a exceção
  - Capturar e tratar a exceção
  - Capturar a exceção e lançar uma exceção que está lista na sua cláusula `throw`
  - Declarar a exceção em sua cláusula `throws` e não tratar a exceção

Tratamento de exceções em Java

- A cláusula `finally`
  - Pode aparecer no final da construção `try`
  - O propósito da cláusula `finally` é especificar código que deve ser executada independente do que acontece na cláusula `try`

Tratamento de exceções em Java

- Arquivo `GradeDistribution.java`
- Arquivo `GradeDistribution2.java` (uso adequado de exceção)

Exemplo em Python

- Separação da leitura e processamento usando geradores
- Arquivo `grade_distribution.py`

Tratamento de exceções em Java

- Avaliação
  - Os tipos das exceções fazem mais sentido do que em C++
  - A cláusula `throws` é melhor do que a do C++
  - A cláusula `finally` é bastante útil
  - A JVM pode gerar uma variedade de exceções que podem ser manipuladas pelo programa do usuário

Introdução a tratamento de eventos

Introdução a tratamento de eventos

- Um **evento** é criado por uma ação externa, como por exemplo, a interação do usuário com uma interface gráfica
- Um **tratador de evento** é um segmento de código que é chamado em resposta a um evento

Tratamento de eventos em Java

Tratamento de eventos em Java

- Os tratadores de eventos são chamados de **ouvidores de evento** (event listeners) em Java
- Um gerador de evento avisa que um evento ocorreu através do envio de mensagem (chamada de método)
- Uma interface é usada para definir um protocolo para os métodos que tratam os eventos
- Os ouvidores devem ser registrados nos geradores de evento explicitamente

Tratamento de eventos em Java

- Exemplo
  - Um tipo de evento é o `ItemEvent`, que representa o evento de clicar em um checkbox, radio button ou item de lista
  - A interface `ItemListener` define o método `itemStateChanged`, que manipula os eventos do tipo `ItemEvent`
  - Os ouvidores são registrados com o método `addItemListener`

Referências

- Robert Sebesta, Concepts of programming languages, 9ª edição. Capítulo 14.