Paradigma Imperativo

Controle de Exceção

ERICK GALANI MAZIERO
erick.maziero@ufla.br

Departamento de Ciências da Computação
Universidade Federal da Lavras

UNIVERSIDADE FEDERAL DE LAVRAS

6

Controle de Exceção

Controle de Exceção

Projetistas de linguagens modernas incluem mecanismos que permitem que um programa possa responder de uma maneira padrão a erros de execução, bem como outros eventos não-usuais.

Em geral, o suporte ao controle de exceção não é uma habilidade simples de ser adicionada à linguagem, por isso não é um recurso ainda amplamente suportado.

Controle de Exceção

Exceções podem ser entendidas como erros detectados por hardware ou eventos não-usuais de execução (ex: fim de arquivo ou ausência de permissão) detectável por software ou hardware e que pode requerer tratamento especial.

O processamento especial requerido para quando ocorre uma exceção é denominado geralmente de **tratamento ou controle de exceção**.

Controle de Exceção

A ausência de suporte a tratamento de exceções não implica que a linguagem não possibilite tratamento de erros. Em C, funções que podem gerar erros a serem tratados possuem retorno para verificação (ex: funções para abertura ou escrita em arquivos).

Em C++

```
try {
	// código que pode gerar uma exceção
}
catch (exception& e) {
	// tratamento da exceção
	cerr << "exceção capturada: " <<
e.what();
}
```

Gera-se exceção por meio de `throw(exceção)`.

Em Java

```
try {
        // código que pode gerar uma exceção
}
catch (FileNotFoundException e1) {
        // tratamento de arquivo não encontrado
}
catch (Exception e2) {
    // tratamento de outras exceções
}
```

Gera-se exceção por meio de `throw(exceção)`.

Em Java

```
try {
	// código que pode gerar uma exceção
}
catch (Exception e) {
	// tratamento da exceção
}
finally {
	// Esse bloco de código será executado independentemente da ocorrência de erros
}
```

Em Python

```
try:
        # código que pode gerar uma exceção
except:  # para qualquer exceção
        # tratamento da exceção
finally:
        # sempre é executado
```

Em Python

```
try:
      print(x)
except NameError:  # para exceção específica
      print(”Variável x não definida”)
except: # para qualquer outra exceção
      # tratamento
finally:
      # sempre é executado
```

Em Python

```
try:
        a = float(input())
except ValueError:
        print(”Erro na conversão para float”)
```

Tratamento de eventos

Similar ao tratamento de exceções, em ambos os tratadores são implicitamente chamados pela ocorrência de algo: erro (exceção) ou evento.

Eventos são gerados por ações externas ao software:

- Interações de um usuário em uma interface gráfica (GUI)
- Partes do código são executadas em momentos imprevisíveis

Tratador de eventos

Segmento de código executado em reposta a um evento

Habilitam subprogramas a responder às interações do usuário (eventos)

Eventos em Java

GUI do Java Swing (*javax.swing*)

```
...
JButton botao = new JButton("Botão");
botao.addActionListener(this);
...
public void actionPerformed(ActionEvent
event) {
button.setText("O botao foi apertado!!!"); }
```

Criação do elemento gráfico (botão) que 'ativa' um evento

Função que tratará o evento 'clicar no botão'. O texto do botão será alterado

Referência Bibliográfica

Sebesta, R. W. (2011). *Conceitos de Linguagens de Programação*. 9 ed. Bookman.

Capítulo 14