8. UMA INTRODUÇÃO AO TRATAMENTO DE EXCEÇÕES

Nesta seção é apresentada uma introdução ao tratamento de exceções, que permite gerenciar os erros durante a execução de uma forma organizada. Assim, pode-se invocar uma rotina de tratamento de erro quando um erro ocorrer, pois o tratamento de fornece um meio de transferir o controle e as informações de um ponto na execução de um programa para um "tratador de exceções" associado a um ponto previamente passado pela execução. Um tratamento de exceções pode ser usado para suportar noções de tratamento de erros e computação tolerante a falhas [2, 10].

O mecanismo de tratamento de exceções requer o uso de três palavras chave: ***try***, ***catch*** e ***throw***. Nos termos mais gerais, os comandos de programa que devem ser monitorados para as exceções estão contidos em um bloco de prova (*try*). Se uma exceção (ou erro) ocorrer dentro do bloco de prova, será disparada usando *throw*. A exceção é pega, usando *catch*, e processada.

Resumindo: qualquer comando que dispara uma exceção precisa ter sido executado dentro de um bloco *try*. Qualquer exceção precisa ser pega por um comando *catch* que segue imediatamente o comando *try* que dispara a exceção, isto é, se uma exceção é chamada no bloco *try*, o controle do programa é transferido para o gerenciador de exceção apropriado. É importante citar que as funções chamadas a partir de dentro de um bloco *try* também podem disparar uma exceção, e que o comando *catch* irá gerenciar qualquer exceção, independente do tipo, como mostra o exemplo abaixo:

```
try {
   // Bloco de prova - inclui qualquer código que pode chamar uma exceção (throw)
}
catch (tipo1 arg) {
  // Executa algumas ações
}
catch (tipo2 arg) {
  // Executa algumas ações
}
:
catch (tipoN arg) {
  // Executa algumas ações
}
```

O bloco de prova pode ser pequeno, tendo somente alguns comandos dentro de uma função, ou pode englobar tudo, envolvendo o código da função *main()* dentro de um bloco *try* (o que efetivamente faz o programa inteiro ser monitorado). Quando uma exceção é disparada, ela é pega pelo comando *catch* correspondente, que processa a exceção. Pode haver mais de um comando *catch* associado com um *try*. Qual comando *catch* será usado é determinado pelo tipo da exceção. Isto é, se o tipo de dado especificado por um *catch* for igual ao da exceção, esse comando *catch* será executado e todos os outros serão ignorados. Quando uma exceção for pega, "*arg*" receberá seu valor. Qualquer tipo de dado pode ser pego, incluindo as classes criadas. Se nenhuma exceção for disparada, isto é, nenhum erro ocorrer dentro do bloco de prova, nenhum comando *catch* será executado.

O tratamento de exceções só será invocado por uma expressão *throw* chamada no código executado no bloco *try* ou em funções chamadas a partir do bloco *try*. A forma geral do comando *throw* é a seguinte:

```
throw <exceção>;
```

throw precisa ser executada ou a partir de dentro do bloco de prova, o que lhe é próprio, ou de alguma função chamada (direta ou indiretamente) a partir de dentro do bloco de prova. "<exceção>" é o valor disparado. Caso seja disparada uma exceção para a qual não existe um comando *catch* aplicável, um encerramento anormal do programa poderá ocorrer. Disparar uma exceção não tratada faz a função "*terminate()*" ser chamada, que, por padrão, chama "*abort()*" para parar o programa.

O exemplo a seguir mostra o modo como o tratamento de exceções de C++ funciona:

```
// Um exemplo simples de tratamento de exceções
#include <iostream>
using namespace std;
int main () {
	cout << "Inicio \n";
	try {				// inicia um bloco de prova
		cout <<  "Dentro do bloco de prova \n";
		throw 100;		// disparou um erro, execução do programa é transferida para a expressão catch
		cout << "Isto nao sera executado!";
	}
	catch (int i) {			// pega um erro, isto é, processa uma exceção de inteiros
		cout << "Pegou uma excecao - o valor e': ";
		cout << i << "\n";
	}
	cout << "Fim";
	return 0;
}
```

Este programa exibe a seguinte saída:

```
Inicio
Dentro do bloco de prova
Pegou uma excecao - o valor e': 100
Fim
```

Normalmente, o código dentro de um comando *catch* tenta reparar um erro tomando uma ação apropriada. Se o erro puder ser corrigido, a execução continuará com os comandos que seguem o *catch*. No entanto, às vezes um erro não pode ser corrigido e um bloco *catch* encerrará o programa com uma chamada a "*exit()*" ou "*abort()*".

É importante ressaltar que o tipo da exceção precisa corresponder ao tipo especificado em um comando *catch*. Por exemplo, no programa anterior, se for alterado o tipo no comando *catch* para *double*, a exceção não será pega e um encerramento anormal ocorrerá. Também pode-se ter mais de um *catch* associado com um *try*, mas cada *catch* precisa pegar um tipo diferente de exceção.

Como já comentado, uma exceção pode ser disparada a partir de um comando que esteja fora do bloco de prova, desde que esteja dentro de uma função que é chamada a partir de dentro do bloco de prova. Da mesma forma um bloco de prova *try* pode estar localizado em uma função. Quando isso ocorre, cada vez que a função é iniciada, o tratamento de exceção relativo a essa função é reinicializado.

Na verdade o tratamento das exceções foi criado para fornecer uma meio estruturado pelo qual o programa pode tratar os eventos anormais. Isso implica que a rotina de tratamento de erro precisa fazer alguma coisa racional quando um erro ocorrer. Por exemplo, o programa a seguir recebe dois números e divide o primeiro pelo segundo. Ele usa o tratamento de exceções para tratar o erro da divisão por zero [10].

```
#include <iostream>
using namespace std;
void divide (double a, double b);
int main () {
	double i, j;
	do {
		cout << "Informe o numerador (0 para parar): ";
		cin >> i;
		cout << "Informe o denominador: ";
		cin >> j;
		divide (i, j);
	} while (i != 0);
	return 0;
}
void divide (double a, double b) {
	try {
		if (!b) throw b;		// verifica a divisao por 0
		cout << "Resultado: " << a/b << "\n";
	}
	catch (double b) {
		cout << "Nao pode ser efetuada uma divisao por zero. \n";
	}
}
```