Templates em C++

Lançamento de Exceção em C++

Introdução

Lançar exceções em C++ é uma forma de sinalizar que ocorreu um erro ou uma condição excepcional. O mecanismo de exceções permite que você separe o código de tratamento de erros do código principal, tornando o programa mais legível e fácil de manter.

1. Definição e Sintaxe

- Definição: O lançamento de exceções é feito usando a palavra-chave throw, seguida por um objeto que representa a exceção.
- Sintaxe:

```
throw expressão;
```

2. Lançamento de Exceção com Tipos Padrão

- Definição: Você pode lançar exceções usando tipos padrão como std::exception, std::runtime_error, std::logic_error, etc.
- Exemplo:

```
int main() {
    try {
        throw std::runtime_error("Erro de tempo de execução");
    } catch (const std::runtime_error& e) {
```

```
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

3. Lançamento de Exceções Personalizadas

- Definição: Você pode definir e lançar suas próprias exceções derivando da classe std::exception.
- Exemplo:

```
class MinhaExcecao : public std::exception {
public:
    const char* what() const noexcept override {
        return "Minha exceção personalizada";
    }
};

int main() {
    try {
        throw MinhaExcecao();
    } catch (const MinhaExcecao& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

4. Lançamento Condicional de Exceções

- Definição: Você pode lançar exceções com base em condições específicas dentro do seu código.
- Exemplo:

```
void validarIdade(int idade) {
    if (idade < 0) {
        throw std::invalid_argument("Idade não pode ser negativa");
    }
}

int main() {
    try {
        validarIdade(-1);
    } catch (const std::invalid_argument& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

5. Lançamento de Exceções em Funções

- Definição: Exceções podem ser lançadas a partir de funções e capturadas no nível da chamada.
- Exemplo:

```
void funcao() {
    throw std::runtime_error("Erro na função");
```

```
}

int main() {
    try {
        funcao();
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

6. Re-lançamento de Exceções

- Definição: Você pode re-lançar uma exceção capturada usando a palavra-chave throw sem argumentos dentro de um bloco catch.
- Exemplo:

```
int main() {
    try {
        try {
            throw std::runtime_error("Erro original");
        } catch (const std::runtime_error& e) {
            std::cout << "Tratando e re-lançando: " << e.what() << std::endl;
            throw;  // Re-lança a exceção
        }
    } catch (const std::runtime_error& e) {
```

```
        std::cout << "Exceção re-lançada capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

7. Propagação de Exceções

- Definição: Exceções podem ser propagadas através da pilha de chamadas até serem capturadas por um bloco catch.
- Exemplo:

```
void funcaoA() {
    throw std::runtime_error("Erro em funcaoA");
}

void funcaoB() {
    funcaoA();
}

int main() {
    try {
        funcaoB();
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
```

```
}
```

8. Lançamento de Exceções em Construtores

- Definição: Exceções podem ser lançadas a partir de construtores para indicar falhas na inicialização de objetos.
- Exemplo:

```
class Exemplo {
public:
    Exemplo() {
        throw std::runtime_error("Erro no construtor");
    }
};

int main() {
    try {
        Exemplo e;
    } catch (const std::runtime_error& e) {
        std::cout << "Exceção capturada: " << e.what() << std::endl;
    }
    return 0;
}
```

Dicas de Boas Práticas

- Lançamento Apropriado de Exceções: Use exceções para condições de erro e situações excepcionais, não para controle de fluxo regular.
- Clareza e Manutenção: Mantenha o código de lançamento de exceções claro e bem documentado para facilitar a leitura e a manutenção do código.
- Tratamento Adequado de Exceções: Certifique-se de capturar e tratar exceções de maneira adequada para evitar comportamentos indesejados no programa.

Esta seção abrange os conceitos sobre o lançamento de exceções em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/throw