

UFMG  
UNIVERSIDADE FEDERAL  
DE MINAS GERAIS

## Programação e Desenvolvimento de Software 2

### Tratamento de exceções

Prof. Douglas G. Macharet  
douglas.macharet@dcc.ufmg.br

DCC  
DEPARTAMENTO DE  
CIÊNCIA DA COMPUTAÇÃO

## Introdução

- O que é uma exceção?
  - “Exceptional event”
  - Algum evento/acidente inesperado que ocorre no contexto da execução do programa
- Importante tratar e gerenciar esses eventos
  - Diferente das asserções (erros fatais)
  - Problema que pode não ser somente do código
  - Demandam alteração no fluxo de execução

PDS 2 - Tratamento de exceções 2

## Introdução

- O que pode gerar uma exceção?
  - Entradas inválidas, falhas de hardware, ...
- Exemplos
  - Timeout ao enviar dados pela rede
  - Abrir um arquivo inexistente
  - Acessar uma posição inválida em um vetor
- O que fazer nesses casos?

PDS 2 - Tratamento de exceções 3

PDS 2 - Tratamento de exceções

A problem has been detected and windows has been shut down to prevent damage to your computer.

DRIVER\_IRQL\_NOT\_LESS\_OR\_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup options, and then select Safe Mode.

Technical Information:

\*\*\* STOP: 0x000000D1 (0x0000000C, 0x00000002, 0x00000000, 0xF86B5A89)

\*\*\* gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory  
Physical memory dump complete.  
Contact your system administrator or technical support group for further assistance.

## Introdução


### Tratamento

- Ignorar
  - É um alarme falso, continuar a execução
- Reportar
  - Escrever uma mensagem na tela (arquivo)
- Terminar
  - Interromper completamente a execução
- Reparar
  - Corrigir e tentar se recuperar (prosseguir)

PDS 2 - Tratamento de exceções 5

## Introdução

### Tratamento

- Definir valor de uma variável global
- Convenção de códigos de retorno
  - C/C++: 0 e !0
  - Java: boolean
- Retornar a mesma resposta da vez anterior
  - Retornar o valor válido mais próximo
- Chamar rotina de processamento de erros
- Lançar uma exceção e tratá-la! 

PDS 2 - Tratamento de exceções 6

## Exceções

- Maneira facilitada de informar que a rotina não deve (pode) continuar a execução
- Sinalização da existência de um problema
  - É criada uma variável que representa a falha
  - A exceção deve então ser “lançada”
  - O código é desviado da execução normal
- Tratamento
  - A “captura” da exceção também deve ser feita

DCC 

PDS 2 - Tratamento de exceções

7

## Exceções

- C++
  - Tratamento estruturado (parte da linguagem)
  - Mais poderoso e flexível que códigos de retorno
  - `try-throw-catch`
- Definidas como classes (ou qualquer tipo)
  - Vantagens do paradigma OO
  - Contém informações sobre o erro (contexto)
  - Pré-definidas / Criadas pelo programador

DCC 

PDS 2 - Tratamento de exceções

8

## Exceções

- Observada pela instrução `try`
  - Região protegida (observável)
  - Bloco de código onde pode ocorrer a exceção
- Capturada pela instrução `catch`
  - Bloco específico para cada tipo de exceção
  - Responsável pelo tratamento (manipulação)

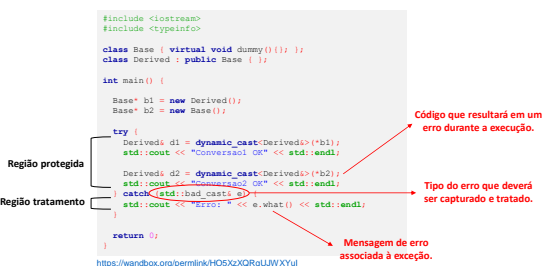
[https://en.cppreference.com/w/cpp/language/try\\_catch](https://en.cppreference.com/w/cpp/language/try_catch)
DCC 

PDS 2 - Tratamento de exceções

9

## Exceções

### Exemplo 1



```

#include <iostream>
#include <typeinfo>

class Base { virtual void dummy() {} };
class Derived : public Base { };

int main() {
    Base* b1 = new Derived();
    Base* b2 = new Base();

    try {
        Derived* d1 = dynamic_cast<Derived>(&b1);
        std::cout << "Conversão OK" << std::endl;

        Derived* d2 = dynamic_cast<Derived>(&b2);
        std::cout << "Conversão OK" << std::endl;
        catch (std::bad_cast&) {
            std::cout << "Error" << e.what() << std::endl;
        }

        return 0;
    }
}

```

Código que resultará em um erro durante a execução.

Tipo do erro que deverá ser capturado e tratado.

Mensagem de erro associada à exceção.

<https://wandbox.org/permlink/HOSXzXQPaUWXYul>

DCC 

PDS 2 - Tratamento de exceções

10

## Exceções

- Lançamento através do operador `throw`
  - Sempre dentro de um bloco `try` (local ou não)
  - Se nada tratar (`catch`), o programa terminará
- A exceção lançada é um objeto
  - Previamente instanciado
  - Instanciado no momento do lançamento
  - Tipo deve ser parâmetro de um bloco `catch`

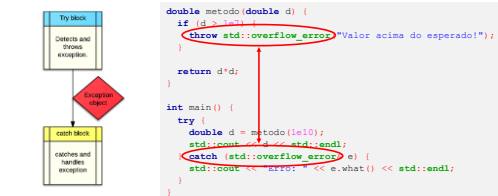
<https://en.cppreference.com/w/cpp/language/throw>
DCC 

PDS 2 - Tratamento de exceções

11

## Exceções

### Exemplo 2



```

double metodo(double d) {
    if (d > 1e3)
        throw std::overflow_error("Valor acima do esperado!");

    return d*d;
}

int main() {
    try {
        double d = metodo(1e4);
        std::cout << d << std::endl;
    } catch (std::overflow_error e) {
        std::cout << "Error: " << e.what() << std::endl;
    }
}

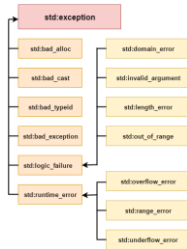
```

DCC 

PDS 2 - Tratamento de exceções

12

## Exceções



DCC

PDS 2 - Tratamento de exceções

13

## Exceções Exemplo 3

```

int main() {
    try {
        // throws std::out_of_range
        std::string("abc").substr(10);
    } catch (std::exception& e) {
        std::cout << e.what() << std::endl;
    }
}

```

[https://en.cppreference.com/w/cpp/string/basic\\_string\\_substr](https://en.cppreference.com/w/cpp/string/basic_string_substr)

DCC

PDS 2 - Tratamento de exceções

14

## Exceções

- Um bloco `try` pode possuir um número qualquer de blocos `catch` associados a ele
- Exceções associadas pelo tipo de exceção
  - Será utilizado o primeiro tratador encontrado cujo parâmetro seja do mesmo tipo da exceção
- É recomendado usar blocos mais específicos

DCC

PDS 2 - Tratamento de exceções

15

## Exceções Exemplo 4

```

#include <vector>

int main() {
    try {
        // throws std::out_of_range
        std::vector<int> v(10);
        v.at(20) = 100;

        // throws std::length_error
        v.resize(v.max_size() + 1);
    } catch (std::out_of_range& e) {
        std::cout << "out_of_range: " << e.what() << std::endl;
    } catch (std::length_error& e) {
        std::cout << "length_error: " << e.what() << std::endl;
    }

    return 0;
}

```

<https://wandbox.org/permlink/DgduA2OvE1K5q>

DCC

PDS 2 - Tratamento de exceções

16

## Exceções Exemplo 5

```

#include <vector>

int main() {
    try {
        // throws std::out_of_range
        std::vector<int> v(10);
        v.at(20) = 100;

        // throws std::length_error
        v.resize(v.max_size() + 1);
    } catch (std::exception& e) {
        std::cout << "exception: " << e.what() << std::endl;
    } catch (std::out_of_range& e) {
        std::cout << "out_of_range: " << e.what() << std::endl;
    } catch (std::length_error& e) {
        std::cout << "length_error: " << e.what() << std::endl;
    }

    return 0;
}

```

<https://wandbox.org/permlink/GRf9XReY7Ra0du>

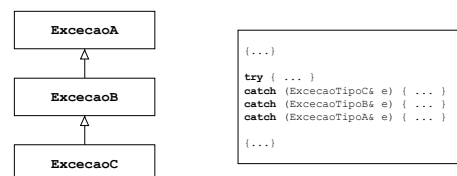
DCC

PDS 2 - Tratamento de exceções

17

## Exceções

- Havendo mais de uma cláusula `catch`, devem estar ordenadas da mais específica para a mais genérica



DCC

PDS 2 - Tratamento de exceções

18

## Exceções

- Pilha de execução
  - Toda invocação é empilhada em uma estrutura de que isola a área de memória de cada método
  - Quando um método termina (retorna), ele volta para o método que o invocou
- Stack Unwinding
  - Procurar primeiro `catch` para tratar a exceção
  - Variáveis no escopo entre o `throw` e o `catch` são destruídas (desenrolamento)

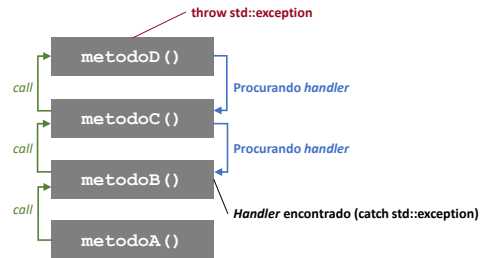
<https://docs.microsoft.com/en-us/cpp/cpp/exceptions-and-stack-unwinding-in-cpp?view=vs-2019>

DCG III

PDS 2 - Tratamento de exceções

19

## Exceções Stack Unwinding



DCG III

PDS 2 - Tratamento de exceções

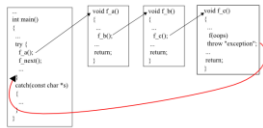
20

## Exceções Stack Unwinding

Sem exceção



Com exceção



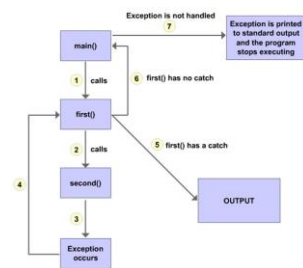
<https://www.bogotobogo.com/cpp/plus/stackunwinding.php>

DCG III

PDS 2 - Tratamento de exceções

21

## Exceções Stack Unwinding



DCG III

PDS 2 - Tratamento de exceções

22

## Exceções Exemplo 6

```
class ClasseB {
public:
    ClasseB() { cout << "Construtor::B" << endl; }
    ~ClasseB() { cout << "Destructor::B" << endl; }
    void metodoB() {
        throw exception();
    }
};

class ClasseA {
public:
    ClasseA() { cout << "Construtor::A" << endl; }
    ~ClasseA() { cout << "Destructor::A" << endl; }
    void metodoA() {
        ClasseB b;
        b.metodoB();
    }
};

int main() {
    try {
        ClasseA a;
        a.metodoA();
    } catch (exception e) {
        cout << e.what() << endl;
    }
}
```

Construtor::A  
Construtor::B  
Destructor::B  
Destructor::A  
std::exception

DCG III

PDS 2 - Tratamento de exceções

23

## Exceções

- Após a exceção o código é desviado
  - Comandos subsequentes não serão executados
  - Problema em algumas situações
    - Fechamento de uma conexão ou arquivo
- Adicionar fluxo que sempre seria executado
  - Suportado em algumas linguagens (`finally`)
  - C++ não possui esse operador → RAII
    - Responsabilidade do programador e não usuário

DCG III

PDS 2 - Tratamento de exceções

24

## Exceções

- É possível criar exceções especializadas
  - Herdar de `std::exception` (ou subclasse)
  - Sempre que possível utilizar as existentes
- Permitem tratamento mais específico
  - Facilitar a identificação dos casos excepcionais
  - Informações necessárias para o tratamento

## Exceções

### Exemplo 7

```
class ExcecaoSaldoInsuficiente : public std::exception {
public:
    virtual const char* what() const throw() {
        return "Erro: SaldoInsuficiente.";
    }
};

class Conta {
    int _agencia;
    int _numero;
    double _saldo = 0;

private:
    bool possuiSaldoSuficiente(double valor) {
        return (_saldo - valor) > 0;
    }

public:
    void sacar(double valor) {
        if (!possuiSaldoSuficiente(valor)) {
            throw ExcecaoSaldoInsuficiente();
        }
        this->_saldo -= valor;
    }
};
```

## Exceções

### Exemplo 7

```
int main() {
    try {
        Conta c;
        c.sacar(100);
    } catch (ExcecaoSaldoInsuficiente& e) {
        std::cout << e.what() << std::endl;
    }

    return 0;
}
```

<https://wandbox.org/permlink/bdWovGethD5ASJ>

## Exceções

- Propagação (rethrow)
  - Quando não se deseja (ou não se sabe) tratar uma certa exceção em um determinado escopo
  - “Relançar” a exceção capturada
- Chamar novamente `throw`
  - Não passar parâmetros (mesmo tipo do `catch`)
  - Podem ser passados parâmetros, mas cuidado!
    - Comportamentos inesperados (herança)
    - Novos objetos de exceção gerados (cópia)

<https://www.learncpp.com/cpp-tutorial/14-6-rethrowing-exceptions/>

## Exceções

### Exemplo 8

```
class MyException : public std::exception {
public:
    virtual const char* what() const throw() {
        return "Erro: MyException.";
    }
};

void metodoTeste() {
    try {
        throw MyException();
    } catch (MyException& e) {
        std::cout << "Dentro -> " << e.what() << std::endl;
        throw;
    }
}
```

Propaga a exceção. Pode ser capturada novamente em outro catch.

## Exceções

### Exemplo 8

```
int main() {
    try {
        metodoTeste();
    } catch (MyException& e) {
        std::cout << "Fora -> " << e.what() << std::endl;
    }

    return 0;
}
```

<https://wandbox.org/permlink/tH5YUhwTTH0Ltkd>

## Exceções

### Exemplo 9

```
class MinhaExcecao {
public:
    std::string mensagem() {
        return "Erro: MinhaExcecao.";
    }
};

int main() {
    try {
        throw MinhaExcecao();
    } catch (std::exception e) {
        std::cout << e.what() << std::endl;
    } catch (...) {
        std::cout << "Excecao desconhecida!" << std::endl;
    }

    return 0;
}
```

Tratamento para  
qualquer tipo de  
exceção que ocorra.

DCC

PDS 2 - Tratamento de exceções

31

## Exceções

### Especificações de exceções

- Indicam a intenção do programador sobre os tipos das exceções que podem ser lançadas
- Adicionadas às assinaturas dos métodos
  - C++
    - C++11: `throw()` / `throw(type)`
    - C++17: `noexcept(true)` / `noexcept(false)`
  - Java: `throws`
- Prática não recomendada em C++
  - Java: Verificadas (compilação) vs. Não Verificadas

[https://en.cppreference.com/w/cpp/language/except\\_spec](https://en.cppreference.com/w/cpp/language/except_spec)

DCC

PDS 2 - Tratamento de exceções

32

## Exceções

### Exemplo 10

```
class Conta {
    int _agencia;
    int _numero;
    double _saldo = 0;

private:
    bool possuiSaldoSuficiente(double valor) {
        return (_saldo - valor) > 0;
    }

public:
    void sacar(double valor) throw() {
        if (!possuiSaldoSuficiente(valor)) {
            throw ExcecaoSaldoInsuficiente();
        }
        this->_saldo -= valor;
    }
};
```

Programa encerraria  
abruptamente com a  
chamada da função  
`terminate()`!

DCC

PDS 2 - Tratamento de exceções

33

## Exercício

```
#include <string>
#include <iostream>

std::string pegar_sub_string(std::string str, int k) {
    return str.substr(k);
}

std::string ler_entrada() {
    std::string texto;
    std::cin >> texto;
    return pegar_sub_string(texto, 10);
}

int main() {
    std::cout << ler_entrada();
    return 0;
}
```

DCC

PDS 2 - Tratamento de exceções

34

## Exercício

```
std::string ler_entrada() {
    std::string texto;
    try {
        std::cin >> texto;
        return pegar_sub_string(texto, 10);
    } catch (std::out_of_range e) {
        std::cerr << "Entrada invalida!" << std::endl;
        return "";
    }
}
```

DCC

PDS 2 - Tratamento de exceções

35

## Exercício

```
std::string ler_entrada() {
    std::string texto;
    while (1) {
        try {
            std::cin >> texto;
            return pegar_sub_string(texto, 10);
        } catch (std::out_of_range e) {
            std::cerr << "Entrada invalida! Digite novamente.\n";
        }
    }
}
```

<https://wandbox.org/permink/W0Hx7ZAsbg8BDd>

DCC

PDS 2 - Tratamento de exceções

36

## Considerações finais

- Problemas no tratamento de exceções
  - Misturar (ou favorecer) outros tratamentos
    - Utilização de códigos de erros
  - Não entender o processo de stack unwinding
  - Não utilizar exceções em construtores
    - Única forma de indicar o problema
  - Lançar exceções nos destrutores
    - Pode ocasionar a chamada de `terminate()`
  - Não capturar uma exceção por referência

<https://www.acodersjourney.com/top-15-c-exception-handling-mistakes-avoid/>

CC BY

PDS 2 - Tratamento de exceções

37