Exceções – stdexcept

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

stdexcept

- O *header* stdexcept define diversas exceções padrão que podem ser lançadas
  - Classes específicas que representam exceções específicas
  - Dê uma olhada:
    - www.cplusplus.com/reference/stdexcept
    - www.cplusplus.com/reference/exception/exception/
- **Boa prática**
  - Sempre lance
    - Uma exceção de stdexcept
      - Ou
    - Suas próprias exceções
      - Devem derivar de alguma das classes de stdexcept
  - Toda exceção **deve derivar de std::exception**

Exceções padrão

- Leia sobre a exceção invalid_argument
  - www.cplusplus.com/reference/stdexcept/invalid_argument

Pessoa.cpp

```cpp
#include "Pessoa.hpp"

#include <stdexcept>


//...
void Pessoa::setIdade(const unsigned short int idade){
    if(idade > 150)
        throw std::invalid_argument{"Idade Invalida."};
    this->idade =   idade;
}
```

Argumento inválido se a idade é maior que 150!!!

UDESC

No main

- Como podemos pegar essa exceção no main?

No main

- Como podemos pegar essa exceção no main?

```
//...
int main(){
        Pessoa *p{nullptr};
        std::string nome;
        unsigned long cpf;
        unsigned short int idade;


        std::cout << "Digite o nome: ";
        std::cin >> nome;
        std::cout << "Digite o cpf: ";
        std::cin >> cpf;
        std::cout << "Digite a idade: ";
        std::cin >> idade;
        try{
                p = new Pessoa{nome, cpf};
                p->setIdade(idade);
                std::cout << p->getNome()
                        << " " << p->getCpf()
                        << " " << p->getIdade() << std::endl;
        }catch(std::invalid_argument& iv){
                std::cout << "Argumento inválido: " <<   iv.what() << std::endl;
        }
        delete p;
        return 0;
}
```

UDESC

No main

- Como podemos pegar essa exceção no main?

```
//...
int main(){
        Pessoa *p{nullptr};
        std::string nome;
        unsigned long cpf;
        unsigned short int idade;

        std::cout << "Digite o nome: ";
        std::cin >> nome;
        std::cout << "Digite o cpf: ";
        std::cin >> cpf;
        std::cout << "Digite a idade: ";
        std::cin >> idade;
        try{
                p = new Pessoa{nome, cpf};
                p->setIdade(idade);
                std::cout << p->getNome()
                        << " " << p->getCpf()
                        << " " << p->getIdade() << std::endl;
        }catch(std::invalid_argument& iv){
                std::cout << "Argumento inválido: " <<   iv.what() << std::endl;
        }
        delete p;
        return 0;
}
```

A função **what** retorna a string armazenada para a exceção.

UDESC

Criando nossas próprias exceções

- Vamos criar nossa própria exceção para deixar claros os erros de cpf inválido
- Vamos seguir as **boas práticas**
  - Toda exceção **deve derivar de std::exception**, ou de alguma classe que derive de std::exception em algum momento na hierarquia
- Pesquise sobre a exceção *runtime_error*

CPFInvalidoException.hpp

```
#ifndef CPF_INVALIDO_EXCEPTION
#define CPF_INVALIDO_EXCEPTION

#include <stdexcept>

class CPFInvalidoException : public std::runtime_error{
      public:
            const unsigned long cpf;

            CPFInvalidoException(const unsigned long cpf);
            virtual ~CPFInvalidoException() = default;
};
#endif
```

UDESC

CPFInvalidoException.cpp

```cpp
#include "CPFInvalidoException.hpp"


CPFInvalidoException::CPFInvalidoException(const unsigned long cpf)
    :std::runtime_error{"CPF Invalido."}, cpf{cpf}{
}
```

Pessoa.cpp

- Em Pessoa.cpp, modifique o setCPF

```
//...
void Pessoa::setCpf(const unsigned long cpf){
    if(validarCPF(cpf))
        this->cpf = cpf;
    else
        throw CPFInvalidoException{cpf};
}
```

No main

- Podemos ter **tantos blocos *catch* quanto necessários!**
  - É permitido ter um bloco catch para tratar cada exceção diferente, por exemplo

```
//...
try{
	p = new Pessoa{nome, cpf};
	p->setIdade(idade);
	std::cout << p->getNome()
		<< " " << p->getCpf()
		<< " " << p->getIdade() << std::endl;
}catch(std::invalid_argument& iv){
	std::cout << "Argumento inválido: " <<   iv.what() << std::endl;
}catch(CPFInvalidoException& ci){
	std::cout << "Erro de CPF: " <<   ci.what() << "CPF incorreto: " << ci.cpf << std::endl;
}
```

No main

- Podemos ter **tantos blocos *catch* quanto necessários!**
  - É permitido ter um bloco catch para tratar cada exceção diferente, por exemplo

Note que com a Exceção customizada podemos incluir dados relevantes sobre o erro (nesse caso, o cpf considerado inválido).

```
//...
try{
        p = new Pessoa{nome, cpf};
        p->setIdade(idade);
        std::cout << p->getNome()
                << " " << p->getCpf()
                << " " << p->getIdade() << std::endl;
}catch(std::invalid_argument& iv){
        std::cout << "Argumento inválido: " <<   iv.what() << std::endl;
}catch(CPFInvalidoException& ci){
        std::cout << "Erro de CPF: " <<   ci.what() << "CPF incorreto: " << ci.cpf << std::endl;
}
```

UDESC

Exceções

- Continuaremos explorando o conceito de exceções na próxima aula com
  - Catch de exceções via polimorfismo
  - Dicas de performance
  - Cenários onde exceções podem gerar problemas

Exercícios

- Modifique o exercício da aula passada, para que agora você lance uma exceção padrão ou, onde for necessário, você lance uma exceção customizada criada por você.

Referências

- DEITEL, P.; DEITEL, H. **C++ how to Program**. [S.l.]: Pearson, 2017. ISBN 9780134448237
- STROUSTRUP, B. **The C++ Programming Language**. Pearson Education, 2013. ISBN 9780133522853.