Exceções – Introdução

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Em caso de erro ...

- Em programas estruturados (ex.: em C) é comum as funções retornarem algum código de erro em caso de problemas
  - Exemplo: int dividir(int divisor, int dividendo, int* resultado);
    - A função acima pode retornar 0 caso tudo ocorra bem, ou -1 em caso de divisão por zero
  - Quais erros o programador pode cometer ao usar uma função assim?
  - O que se torna estranho nesse tipo de função?

Em caso de erro ...

- Em programas estruturados (ex.: em C) é comum as funções retornarem algum código de erro em caso de problemas
  - Exemplo: int dividir(int divisor, int dividendo, int* resultado);
    - A função acima pode retornar 0 caso tudo ocorra bem, ou -1 em caso de divisão por zero
  - Quais erros o programador pode cometer ao usar uma função assim?
    - É comum o programador esquecer de verificar o retorno
  - O que se torna estranho nesse tipo de função?
    - No exemplo, o retorno é apenas um flag indicando se tudo correu bem, e o resultado está sendo armazenado em um ponteiro
      - Um tanto confuso

Exceções

- Linguagens orientadas a objetos possuem o conceito de exceção
  - Indicam que algo deu errado
  - Mecanismo padrão para tratamento de erros
    - Criar programas tolerantes a falhas
      - Detectam uma falha em específica e se recuperam delas
    - Detecção de falhas
      - Encontrar a fonte de um problema se torna mais fácil

Exceções

- Uma exceção deve ser **lançada** quando um **problema acontece**
  - Algo inesperado, ou que não deveria acontecer no fluxo normal do programa
- Em C++, podemos lançar **qualquer coisa**
  - Um inteiro, um char, um objeto de uma classe, um ponteiro, …
  - Veremos nas próximas aulas lançar qualquer coisa é uma má prática
    - Mas na aula introdutória de hoje, vamos nos contentar em lançar inteiros

SetCPF

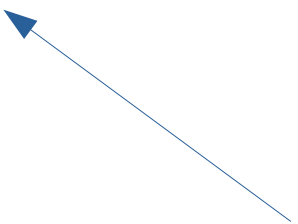
- Abra a classe Pessoa
  - O que pode dar errado no setCpf?

Construtor de Pessoa

- Abra a classe Pessoa
- O que pode dar errado no setCPF?
  - Podemos por exemplo passar um CPF inválido
    - Vamos usar o conceito de exceção para tratar esse caso
- Para lançar uma exceção, utilizamos a palavra-chave **throw**

Construtor de Pessoa

```
void Pessoa::setCpf(const unsigned long cpf){
    if(validarCPF(cpf))
        this->cpf = cpf;
    else
        throw (int)1;
}
```



Caso o CPF seja inválido, lance (throw) o inteiro 1

Ao encontrar um throw

- Quando um throw é executado
  - O objeto/variável/ponteiro **é imediatamente lançado** para que alguém o pegue
  - A função é abortada **imediatamente**
    - Não termina sua execução
    - Não retorna resultados

Para pegar uma exceção

- Para executar um trecho que pode lançar uma exceção
- O trecho deve ficar dentro de um bloco ***try***
- Após o try, adicionamos um bloco ***catch(tipo nomeExceção)***
  - O *catch* “pega” a exceção lançada

Exemplo

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

	std::cout << "Digite o nome: ";
	std::cin >> nome;
	std::cout << "Digite o cpf: ";
	std::cin >> cpf;

	try{
		p = new Pessoa{nome};
		p->setCpf(cpf);
		std::cout << p->getNome() << " " << p->getCpf() << std::endl
	}catch(int& ex){
		if(ex == 1){
			std::cout << "CPF inválido" << std::endl;
		}else{
			std::cout << "Erro desconhecido" << std::endl;
		}
	}
	delete p;
	return 0;
}
```

UDESC

Try-catch

- O try tenta executar o seu bloco
- Se nenhuma exceção for lançada, o try todo é executado
  - O catch não é executado
- Se uma exceção é lançada
  - As instruções do try param de ser executadas **imediatamente**
  - Se o catch espera essa exceção
  - O **catch** todo relacionado a exceção é executado

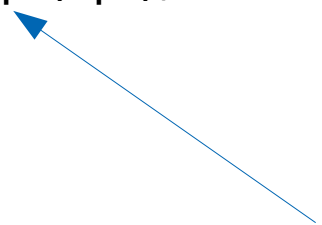
Construtor de Pessoa

- Como podemos melhorar o construtor de Pessoa?

Construtor de Pessoa

- Como podemos melhorar o construtor de Pessoa?

```
Pessoa::Pessoa(const std::string& nome, const unsigned long cpf)
        : Pessoa(nome) {
    this->setCpf(cpf);
}
```



Chamamos a função setCpf

Construtor de Pessoa

- Como podemos melhorar o construtor de Pessoa?
  - Mas e agora, se setCpf lançar uma exceção, quem deveria tratá-la?
    - Em outras palavras, onde deve estar o try-catch?

```
Pessoa::Pessoa(const std::string& nome, const unsigned long cpf)
        : Pessoa(nome) {
    this->setCpf(cpf);
}
```

Construtor de Pessoa

- Como podemos melhorar o construtor de Pessoa?
  - Mas e agora, se setCpf lançar uma exceção, quem deveria tratá-la?
    - Depende: Se o construtor de pessoa puder fazer algo útil quanto a exceção, ele deve pegar a exceção
    - Caso contrário, não pegue a exceção nesse caso
      - A exceção será lançada para quem chamou o construtor

```
Pessoa::Pessoa(const std::string& nome, const unsigned long cpf)
        : Pessoa(nome) {
    this->setCpf(cpf);
}
```

Erro comum

- **Não** faça isso
  - É comum encontrarmos essa construção (errada) em muitos programas
  - Note que temos um *catch*, mas ele não faz nada útil
    - Ele simplesmente pega a exceção, e a relança
    - O mesmo que se não tivéssemos o catch!!!
- Lançar uma exceção e pegá-la no catch custa caro!
  - Se o seu catch não faz algo útil, ele não deveria estar ali

```
Pessoa::Pessoa(const std::string& nome, const unsigned long cpf)
        : Pessoa(nome) {
    try{
        this->setCpf(cpf);
    }catch(int& ex){
        throw ex;
    }
}
```

UDESC

Exceções e construtores

- Quando um construtor lança uma exceção
  - Acontece o mesmo que em uma função qualquer
  - O construtor é abortado imediatamente
    - **O objeto não é construído**

Teste você mesmo

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


        std::cout << "Digite o nome: ";
        std::cin >> nome;
        std::cout << "Digite o cpf: ";
        std::cin >> cpf;


        while(p == nullptr){
                try{
                        p = new Pessoa{nome, cpf};
                        std::cout << p->getNome() << " " << p->getCpf() << std::endl;
                }catch(int& ex){
                        if(ex == 1){
                                std::cout << "CPF inválido, digite outro: " << std::endl;
                                std::cin >> cpf;
                        }else{
                                std::cout << "Erro desconhecido" << std::endl;
                        }
                }
        }
        delete p;
        return 0;
}
```

Exercício

- Localize os demais trechos do programa onde exceções podem ser necessárias, e as lance
  - Por enquanto pode continuar lançando inteiros
  - Nas próximas aulas, discutiremos sobre o lançamento de classes mais sofisticadas e boas práticas

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.