UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

BCC35A - Linguagens de Programação

Prof. Dr. Rodrigo Hübner

Aula 08: Avaliação de expressões, funções de alta ordem, *duck types*, efeito colateral e transparência referencial

Funções de alta ordem

- Quando suportado, funções são tratadas como **valores de primeira classe**
- Funções anônimas (`lambdas`) são funções que não possuem um nome definido e podem ser usadas diretamente como argumentos em chamadas de função ou atribuídas a variáveis
- As funções de alta ordem permitem que você **passe funções como argumentos** para outras funções
- Também podem retornar funções como resultados
- Questões que vimos em linguagens funcionais (`map` , `filter` , etc)

Duck Typing

- Conceito comum em linguagens de programação com tipificação dinâmica, como `Python`, `Ruby` e `JavaScript`
- Se baseia na ideia de que o **tipo de um objeto** é determinado pela **presença dos métodos e propriedades** que são chamados em vez de sua classificação formal

> “ Se um objeto anda como um pato e faz barulho de pato, então é um pato ”

Duck Typing

- O *Duck Typing* permite:
  - **Polimorfismo**: objetos de diferentes classes podem responder...
  - **Interfaces implícitas**: Tem métodos e propriedades necessários, então implementa!
  - **Reutilização de código**: diferentes objetos são utilizados de forma intercambiável em diferentes contextos
  - **Testes baseados em comportamento**: não quero saber o tipo!

Efeitos colaterais

- Ocorre quando uma **função** ou **expressão modifica o estado de algum objeto** externo ou do sistema como um todo.
- Podemos definir um efeito colateral "muito simples" em `C`:

```c
void soma(int a, int b) {
    int resultado = a + b;
    printf("A soma é %d\n", resultado);
    return resultado;

int main() {
  soma(2, 3);
  return 0;
}
```

Expressões em curto-circuito

- Ocorre quando o resultado de uma expressão lógica pode ser determinado com base apenas na **avaliação de uma parte da expressão**, sem a necessidade de avaliar a expressão inteira
- Linguagens que suportam curto-circuito, existem dois operadores lógicos principais: o operador "`and`" (`&&`) e o operador "`or`" (`||`)
- Pode ser útil para melhorar a eficiência e evitar avaliações desnecessárias, especialmente quando a segunda parte da expressão envolve operações custosas ou com efeitos colaterais.

Expressões em curto-circuito

- Vamos analisar o seguinte código em `Javascript`:

```
let a = 5;
let b = 0;

if (a > 0 && (10 / b) > 0) {
    console.log("Expressão verdadeira");
} else {
    console.log("Expressão falsa");
}
```

Expressões em curto-circuito

- O código a seguir em `C` possui efeito colateral?

```c
#include <stdio.h>
int func1() {
    printf("Executando função 1\n"); return 0;
}
int func2() {
    printf("Executando função 2\n"); return 1;
}
int main() {
    if (func1() && func2())
      printf("Expressão verdadeira\n");
    else
      printf("Expressão falsa\n");
}
```

Precedência de operadores

- Qual o resultado da avaliação da expressão `3 + 4 * 5?`. Depende das regras de precedência!
- As regras de **precedência de operadores** para avaliação de expressões definem a ordem que os operadores com diferentes níveis de precedência devem ser avaliados

	Ruby	Like C	Ada
Mais alta	**	pós-fixo ++, --	**, abs
	unário +, -	pré-fixo ++, --, unário +, -	*, /, mod, rem
	*, /, %	*, /, %	unário +, -
Mais baixa	binário +, -	binário +, -	binário +, -

Regras de associatividade

- Determinam a **ordem** na qual **operadores com a mesma precedência são avaliados** quando aparecem em sequência
- **Associatividade** pode ocorrer da direita para a esquerda ou o contrário!
- Múltiplas atribuições ocorre **quase** sempre da direita p/ esquerda:

```
my $a;
my $b;
my $c = 5;
$a = $b = $c;
print "a: $a\n";  # Resultado: a: 5
print "b: $b\n";  # Resultado: b: 5
```

Transparência Referencial

- É a propriedade de uma **função** ou expressão retornar sempre o **mesmo resultado** quando chamada com os **mesmos argumentos**
- Uma **função puramente referencial não tem efeitos colaterais** e **não depende de variáveis globais** ou de estado externo para produzir seu resultado

Próxima aula

- *Bindings* e *Wrappers*