Respostas - Atividades Práticas de Laboratório (Capítulos 5 e 6)

Leonardo Leitao Souza – 23020085-2 – ESOFT6SNA

Capítulo 5 – Nomes, Vinculações e Escopos

Atividade 1 – Escopo Estático x Escopo Dinâmico

Em Python:

```
x = 10
def f():
    print(x)
def g():
    x = 20
    f()
g()  # Saída: 10
```

Em JavaScript:

```
let x = 10;
function f() {
    console.log(x);
}
function g() {
    let x = 20;
    f();
}
g();  // Saída: 10
```

• O valor impresso depende do local de definição da função (escopo estático), não do local de chamada.
• Tanto Python quanto JavaScript utilizam escopo estático (léxico). Nenhuma dessas linguagens usa escopo dinâmico por padrão.

Atividade 2 – Tempo de Vida das Variáveis

Em C:

```
void contador() {
    int a = 0;
    static int b = 0;
    a++;
    b++;
```

```
    printf("a = %d, b = %d\n", a, b);
}

Chamadas sucessivas:
contador(); // a=1, b=1
contador(); // a=1, b=2
contador(); // a=1, b=3
```

• A variável automática 'a' é recriada a cada chamada, sempre reiniciando do zero.
• A variável estática 'b' mantém seu valor entre chamadas, pois seu tempo de vida é o da execução completa do programa.
• Isso demonstra a diferença entre tempo de vida automático (pilha) e estático (memória global).

Capítulo 6 – Tipos de Dados

Atividade 3 – Declaração de Tipos e Coerção

Em Java:

```
int num = 10;
num = "dez"; // Erro de compilação.
```

Em Python:

```
num = 10
num = 'dez'
print(num + 5)  # Erro em tempo de execução: não é possível somar string e int.
```

• O Java não permite a atribuição de string a um inteiro devido à tipagem estática.
• O Python permite, mas pode gerar erros em tempo de execução (tipagem dinâmica).
• Vantagens: Tipagem estática oferece mais segurança e previsibilidade; tipagem dinâmica dá mais flexibilidade e rapidez no desenvolvimento.

Atividade 4 – Trabalhando com Arrays e Registros (Structs)

Em C (array):

```
int numeros[5] = {1, 2, 3, 4, 5};
```

Em C (struct):

```
struct Livro {
    char titulo[50];
    char autor[50];
    int anoPublicacao;
};
struct Livro l1 = {"Livro A", "Autor A", 2020};
```

Em Java:

```
class Livro {
    String titulo;
    String autor;
    int anoPublicacao;
    Livro(String t, String a, int ano) { titulo = t; autor = a; anoPublicacao = ano; }
}

ArrayList<Livro> lista = new ArrayList<>();
lista.add(new Livro("Livro A", "Autor A", 2020));
lista.add(new Livro("Livro B", "Autor B", 2021));
lista.add(new Livro("Livro C", "Autor C", 2022));

for (Livro l : lista) {
    System.out.println(l.titulo);
}
```

• Arrays são estruturas homogêneas (mesmo tipo).
• Structs/Classes permitem reunir diferentes tipos em uma única entidade.
• Arrays são ideais quando precisamos de coleções de elementos simples.
• Structs/Classes são melhores quando representamos entidades complexas com múltiplos atributos.