Capítulo 5 – Nomes, Vinculações e Escopos

Atividade 1 – Escopo Estático x Escopo Dinâmico

codigo py: x = 10

```
def f():
    print(x)
def g():
    x = 20
    f()
g()
```

codigo java: var x = 10;

```
function f() {
    console.log(x);
}
function g() {
    var x = 20;
    f();
}
g();
```

- O valor impresso depende do local de definição ou do local de chamada da função?

No escopo estático, depende do local de definição (resolução léxica no código fonte). No escopo dinâmico, depende do local de chamada (pilha de execução no runtime).

- Qual linguagem se comporta como escopo estático e qual se aproxima de dinâmico?

Python se comporta como escopo estático. JavaScript se aproxima de dinâmico se não declararmos variáveis locais (ex.: omitindo var/let), pois pode modificar o escopo superior, mas isso não é escopo dinâmico verdadeiro (é mais uma peculiaridade de hoisting e globals).

Atividade 2 – Tempo de Vida das Variáveis

codigo: #include <stdio.h>

```
void contador() {
    int a = 0;      // Automática
    static int b = 0; // Estática
    a++;
    b++;
    printf("a: %d, b: %d\n", a, b);
}
```

```
int main() {
   contador();
   contador();
   contador();
   return 0;
}
```

- Por que a sempre reinicia do zero, mas b acumula valor entre chamadas?

A variável a é automática (alocada na pilha), reinicializada a cada chamada da função. A variável b é estática (alocada no segmento de dados), inicializada apenas uma vez e mantém seu valor entre chamadas.

- Como isso se relaciona com tempo de vida da variável?

O tempo de vida das variáveis automáticas é limitado ao escopo da função (destruídas ao sair). O tempo de vida das variáveis estáticas é o do programa inteiro, persistindo valores entre execuções do escopo.

Capítulo 6 – Tipos de Dados

Atividade 3 – Declaração de Tipos e Coerção

```
codigo: num = 10
print(num + 5)
num = 'dez'
try:
   print(num + 5)
except TypeError as e:
   print(str(e))
```

- Por que o Java não permite essa operação e o Python permite?

Java tem tipagem estática (verificação de tipos em tempo de compilação), não permitindo mudança de tipo após declaração. Python tem tipagem dinâmica (verificação em runtime), permitindo reatribuição de tipos diferentes, mas operações incompatíveis falham durante a execução.

- Quais as vantagens e desvantagens de cada abordagem?

Tipagem estática (Java): Vantagens incluem detecção precoce de erros, melhor desempenho e autocompletar em IDEs; desvantagens incluem menos flexibilidade e código mais verboso. Tipagem dinâmica (Python): Vantagens incluem código mais conciso e flexível; desvantagens incluem erros só descobertos em runtime e potencial impacto no desempenho.

Atividade 4 – Trabalhando com Arrays e Registros (Structs)

códigos em c:

```
int arr[5] = {1, 2, 3, 4, 5};
#include <stdio.h>
#include <string.h>

struct Livro {
```

```
    char titulo[100];
    char autor[100];
    int anoPublicacao;
};

int main() {
    struct Livro livro;
    strcpy(livro.titulo, "Conceitos de Linguagens");
    strcpy(livro.autor, "Robert W. Sebesta");
    livro.anoPublicacao = 2016;
    printf("Título: %s\n", livro.titulo);
    return 0;
}
```

código em java:

```
import java.util.ArrayList;

class Livro {
    String titulo;
    String autor;
    int anoPublicacao;

    public Livro(String titulo, String autor, int anoPublicacao) {
        this.titulo = titulo;
        this.autor = autor;
        this.anoPublicacao = anoPublicacao;
    }
}

public class Main {
    public static void main(String[] args) {
        ArrayList<Livro> livros = new ArrayList<>();
        livros.add(new Livro("Livro1", "Autor1", 2001));
        livros.add(new Livro("Livro2", "Autor2", 2002));
        livros.add(new Livro("Livro3", "Autor3", 2003));
```

```
        for (Livro livro : livros) {
            System.out.println(livro.titulo);
        }
    }
}
```

- Qual a diferença entre um array e um registro/objeto?

Um array é um tipo homogêneo (todos os elementos do mesmo tipo, acessados por índice). Um registro (struct) ou objeto é heterogêneo (campos de tipos diferentes, acessados por nome).

- Quando seria mais adequado usar array e quando usar struct/classe?

Use array para coleções de itens semelhantes (ex.: lista de números ou strings). Use struct/classe para representar entidades complexas com atributos variados (ex.: um livro com título, autor e ano).