**Semelhança de uma Struct C e uma classe Java com seus atributos**

Quem já programou em C e em Java sabe que embora as linguagens possuam paradigmas diferentes elas apresentam algumas semelhanças em funcionalidades.

Podemos destacar e fazer um comparativo de uma Estrutura ou Struct C e a simples declaração de uma classe Java e seus atributos.

Uma Struct tem a finalidade de unir em conjunto diversas variáveis de tipos diferentes sob o mesmo nome. Isso facilita muito a vida do programador que ao manipular dados, como por exemplo na criação de um cadastro, não necessita criar um número X vezes a quantidade de variáveis comuns a todos os cadastros.

Supondo que o usuário precise cadastrar 20 funcionários novos com nome, idade, renda, rua e cargo. Sem a Estrutura teríamos que criar algo assim:

**char** nome1 [50], nome2 [50], nome3 [50] ... nome20 [50];

**int** idade1, idade2, idade3 ... idade20;

...

**char** cargo1 [30], cargo2 [30], cargo3 [30] ... cargo20 [30];

Com tantas variáveis a probabilidade de erro ao programarmos é enorme, principalmente se houver atualizações no código. Contando também o esforço e a perda de tempo com cadastros de 100, 1000 novos funcionários.

Usando o conceito de Struct podemos simplificar a programação criando diversos cadastros com apenas uma declaração desta forma:

**struct cadastro** {

**char** nome[30];

**int** idade;

**float** renda;

**char** rua[50];

**char** cargo[50];

} cad1, cad2, cad3 ... cad20; //reticências são para simplificar a

explicação, não fazem parte de um código real.

Podemos facilitar mais ainda usando o conceito de iteração através de um Array de Structs:

**int** main(){

**struct** cadastro cad[20]; // *struct previamente declarada antes da main()*

**int** i;

for(i = 0; i < 20; i++){

gets( cad [i].nome);

scanf(“ %d” , & cad [i].idade);

scanf(“ % f “, & cad[i].renda);

gets( cad[i].rua);

gets( cad[i].cargo);

}

system(“pause”);

return 0;

}

Vejamos agora alguns conceitos em Java e sua estrutura.

Em Java, quando queremos criar programas como cadastros conforme o exemplo anteriormente dado, dizemos que esse cadastro é um objeto de uma classe Cadastro que contém atributos que irão abrigar os dados cadastrais como nome, idade, cargo... etc. .

Veja no exemplo:

**public class** Cadastro{

**public String** nome;

**public int** idade;

**public double** renda;

**public String** rua;

**public String** cargo;

}

A partir da classe Cadastro podemos criar objetos deste tipo e instanciarmos os atributos desejados em todo o programa, seja de forma direta ou através do que chamamos de métodos.

Cadastro cad1 = **new** Cadastro();

Cadastro cad2 = **new** Cadastro();

Cadastro cad3 = **new** Cadastro();

cad1.nome(“Pablo”);

cad1.idade(22);

cad1.renda(2200,0);

cad2.nome(“JOAO”);

cad2.idade(30);

Podemos concluir que independente da linguagem e do paradigma, sempre haverá semelhanças estruturais de linguagem. O que define a usabilidade de uma linguagem de programação é o problema a ser resolvido, observando então se a orientação a objetos ou uma linguagem estruturada se aplica por exemplo. Vimos que ambas as linguagens, C e Java, apresentam soluções estruturais diferentes, mas com conceitos semelhantes e resultados iguais para o problema em questão.

A base da disciplina e dos conhecimentos de algoritmos é praticamente a mesma para diversas linguagens, tendo em particular apenas, aplicada como regra, a sintaxe da linguagem utilizada.

Referências:

Backes, André Liguagem C: completa e descomplicada / André Backes. – Rio de Janeiro: Elsevier, 2013.

Local DevMedia, Site <https://www.devmedia.com.br/programacao-orientada-a-objetos-com-java/18449> : Acesso em 09/10/2021.