UNIVERSIDADE FEDERAL DE SÃO PAULO

Programação Orientada a Objetos

Atividade da aula do dia 03/09:

Nessa atividade, o intuito era criar uma classe Estudante que contivesse nome e notas (array). Criar o array de estudantes, adicionar as notas e imprimir as médias.

Além disso, foi solicitado que fosse feita uma comparação com o C: comparando array de structs vs classes, funções vs métodos.

Para cumprir com todos os requisitos, tomei a iniciativa de além da entrega do código em TypeScript, formular um código semelhante em C, para utilizar como apoio para a comparação entre as linguagens.

Confira os códigos:

- Código em TypeScript

```
class Student {
    name: string
    grades: number[]

    constructor(name: string, grades: number[]) {
        this.name = name
        this.grades = []
    }

    addGrade(num: number): void {
        this.grades.push(num);
    }

    getAverage(grades: number[]) {
        let i: number = 0
        let average: number = 0
        let sum: number = 0

        for (const num of grades) {
            sum += num
            i++
        }

        average = sum / i

        console.log(`A média de ${this.name} foi ${average}.`)
    }
}

const students: Student[] = [
```

```
    new Student("Alice", []),
    new Student("João", [])
];

students[0]?.addGrade(10)
students[0]?.addGrade(9)
students[0]?.addGrade(8)

students[1]?.addGrade(6)
students[1]?.addGrade(7)
students[1]?.addGrade(8)

students[0]?.getAverage(students[0].grades)
students[1]?.getAverage(students[1].grades)
```

- Código em C:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct NameNode
{
    char c;
    struct NameNode *next;
} NameNode;

typedef struct GradeNode
{
    float grade;
    struct GradeNode *next;
} GradeNode;

typedef struct Student
{
    NameNode *name;
    GradeNode *grades;
} Student;

void appendChar(NameNode **head, char c)
{
    NameNode *newNode = (NameNode *)malloc(sizeof(NameNode));
    newNode->c = c;
    newNode->next = NULL;

    if (*head == NULL)
    {
        *head = newNode;
    }
    else
    {
        NameNode *temp = *head;
        while (temp->next != NULL)
```

```
            temp = temp->next;
        temp->next = newNode;
    }
}

void appendGrade(GradeNode **head, float g)
{
    GradeNode *newNode = (GradeNode *)malloc(sizeof(GradeNode));
    newNode->grade = g;
    newNode->next = NULL;

    if (*head == NULL)
    {
        *head = newNode;
    }
    else
    {
        GradeNode *temp = *head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
}

void printStudent(Student s, int index)
{
    printf("Student %d: ", index + 1);
    NameNode *nNode = s.name;
    while (nNode != NULL)
    {
        printf("%c", nNode->c);
        nNode = nNode->next;
    }

    printf("\nGrades: ");
    GradeNode *gNode = s.grades;
    while (gNode != NULL)
    {
        printf("%.1f ", gNode->grade);
        gNode = gNode->next;
    }
    printf("\n\n");
}

float getAverage(GradeNode *grades)
{
    if (grades == NULL)
        return 0.0;

    float sum = 0;
    int count = 0;
    GradeNode *current = grades;

    while (current != NULL)
```

```
    {
        sum += current->grade;
        count++;
        current = current->next;
    }

    return (count > 0) ? (sum / count) : 0.0;
}

int main()
{
    int n = 2;
    Student students[n];

    for (int i = 0; i < n; i++)
    {
        students[i].name = NULL;
        students[i].grades = NULL;
    }

    char *name0 = "John";
    for (int i = 0; name0[i] != '\0'; i++)
        appendChar(&students[0].name, name0[i]);
    float grades0[] = {8.5, 9.0, 7.5};
    for (int i = 0; i < 3; i++)
        appendGrade(&students[0].grades, grades0[i]);

    char *name1 = "Alice";
    for (int i = 0; name1[i] != '\0'; i++)
        appendChar(&students[1].name, name1[i]);
    float grades1[] = {9.5, 8.0, 6.5};
    for (int i = 0; i < 3; i++)
        appendGrade(&students[1].grades, grades1[i]);

    for (int i = 0; i < n; i++)
    {
        printStudent(students[i], i);
        float avg = getAverage(students[i].grades);
        printf("Average: %.2f\n\n", avg);
    }

    return 0;
}
```

Comparação:

Agora, depois de analisar os dois códigos, podemos perceber que existem algumas diferenças cruciais nas lógicas das linguagens, isso se dá pelo fato do TypeScript seguir uma construção baseada em orientação a objetos, enquanto o C se dá como uma linguagem procedural.

Classes em TypeScript são estruturas de alto nível que combinam dados e comportamento. Isto é, um objeto criado a partir de uma classe não só contém propriedades da classe, mas também sabe como manipulá-las

por meio dos métodos daquela classe. No caso dessa atividade nós definimos a classe Student com as propriedade name e grades, e criamos os métodos internos da classe: addGrade (que insere uma nota) e getAverage(que retorna a média daquele aluno). Além disso, é interessante destacar que o TS (TypeScript) nos fornece arrays dinâmicos, com gerenciamento automático da memória, tornando a manipulação de dados simples e direta (podemos por exemplo criar um array sem tamanho pré-definido "grades[]" e depois inserir cada nota no método addGrade utilizando o método "push" próprio do TS).

Enquanto isso, em C, array de structs são estruturas puramente de dados, sem comportamentos. Cada struct armazena informações, mas os métodos para manipulá-las devem ser criados em forma de funções que operam sobre essas structs. Além disso, aqui a manipulação de arrays dinâmicos deve ser realizada manualmente, com a utilização de ponteiros para controle de memória.