UNIVERSIDADE FEDERAL DE SANTA CATARINA
CTS – DEC – ARA
ENC – Engenharia de
Computação Disciplina :
Programação II
Professor : Antonio Carlos Sobieranski

Avaliação P1 – Programação Orientada à Objetos
Aluno: Ítalo Manzine Amaral Duarte Garofalo **Matrícula:** 20204027

Avaliação Prova Teórica (PT) – Prazo para Entrega de ambas (PT e PP) as avaliações: 36 horas

Avaliação a ser desenvolvida individualmente, respostas em um PDF avulso com respostas enumeráveis. Respostas ilegíveis ou não passíveis de interpretação serão desconsideradas na correção.

---

1. **(1.0 ponto)** No que tange o Paradigma Orientado à Objetos (POO). Diferencie Classes, Instâncias e Objetos. *(min 4 linhas)*
   - **R:**
     - Classe é uma abstração que descreve um conjunto de objetos semelhantes, isto é, atributos e métodos que resumem as caracteristicas comuns de vários objetos, a classe é que define o que os objetos são e como se comportam.
     - Instância é um objeto criado no programa.
     - Objeto é uma instância de uma classe, a qual lhe são atribuídos valores e tem as características definidas pela classe.

---

2. **(1.5 pontos)** Defina o conceito de encapsulamento na POO ? Exemplifique fornecendo um exemplo código em C++ de sua autoria, no PDF a ser entregue, deixando claro a ocorrência deste.
   - **R:** Encapsulamento vem de encapsular, se trata de um mecanismo que adiciona segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta, caso um dado esteja encapsulado, ele só pode ser acessado através de métodos da própria classe, um exemplo de encapsulamento seria o “private”.
   - **Ex:**

```
5  class adicao
6  {
7    private:
8      int x, y, z;
9
10   public:
11     void soma()
12     {
13       cout << "Digite dois numeros: " << endl;
14       cin >> x >> y;
15       z = x + y;
16       cout << "O resultado da adicao e: " << z << endl;
17     }
18 };
```

---

3. **(1.5 pontos)** Em relação à Herança. Explique o conceito e apresente as suas principais suas vantagens, detalhando-as.
(min 5 linhas)

- **R:** Herança é um mecanismo que permite classes reutilizarem (herdem) características de uma classe mãe para suas classes filhas, alguns de seus principais benefícios são redução no tempo de manutenção, uma vez que basta editar a classe mãe, maior reutilização de código, pois basta incluir a classe mãe, reduz a complexidade de desenvolvimento, pois haverá menos código para edição, reduzindo margem de erro e facilitando futuras implementações e atualizações.

---

4. **(1.5 pontos)** Quanto a questão anterior: Apresente um exemplo de 3 níveis com menos 2 construtores para cada classe (sobrecarga, desconsiderando o *default*). No main, instancie 5 objetos de tipos variados, realizando a chamada do construtor. *(escreva em C++, somente código)*

- **R:**

```cpp
#include <iostream>
#include <string>

using namespace std;

class Atleta
{
  protected:
    string m_nome;
    int m_idade;

  public:
    Atleta() : m_nome(""), m_idade(0){};
    Atleta(string nome) {m_nome = nome;};
    Atleta(string nome, int idade) {m_nome = nome, m_idade = idade;};
};

class Nadador : public Atleta
{
  protected:
    string m_prova;
    bool nadando;

  public:
    Nadador() : m_prova(""), nadando(){};

    Nadador(string nome, int idade, string prova)
    {
      m_nome = nome;
      m_idade = idade;
      m_prova = prova;
    };
    Nadador(string nome, int idade, string prova, bool nadando)
    {
      m_nome = nome;
      m_idade = idade;
      m_prova = prova;
      nadando = true;
    };
};
```

```
41
42  class Velocista : public Nadador
43  {
44    private:
45      float m_velocidadeNado;
46      bool estilo_straightArm;
47
48    public:
49      Velocista() : m_velocidadeNado(), estilo_straightArm(){};
50
51      Velocista(string nome, int idade, string prova, bool nadando, float velocidadeNado)
52      {
53        m_nome = nome;
54        m_idade = idade;
55        m_prova = prova;
56        nadando = true;
57        m_velocidadeNado = velocidadeNado;
58      };
59      Velocista(string nome, int idade, string prova, bool nadando, float velocidadeNado, bool estilo_straightArm)
60      {
61        m_nome = nome;
62        m_idade = idade;
63        m_prova = prova;
64        nadando = true;
65        m_velocidadeNado = velocidadeNado;
66        estilo_straightArm = true;
67      }
68  };
69
70  int main()
71  {
72    Atleta Atleta("Usain");
73    Nadador NadadorA("Phelps", 36, "200m borboleta");
74    Nadador NadadorB("Popov", 49, "100m livre", false);
75    Velocista VelocistaA("Cielo", 33, "50m livre", true, 2.35);
76    Velocista VelocistaB("Italo", 29, "50m livre", true, 2.19);
77    return 0;
78  }
```

---

5. **(2.0 pontos)** O que é Polimorfismo e qual a sua relação com a Herança ? *(min 5 linhas)* Apresente 01 exemplo.
   - **R:** Polimorfismo se refere à habilidade dos objetos, funções e variáveis de assumirem várias formas, possibilitando que métodos diferentes, que estão em diferentes níveis de hierarquia de classes tenham o mesmo nome e apresentem diferentes comportamentos. Em Programação Orientada a Objetos, esse conceito é utilizado para trabalhar com uma hierarquia de classes criados por meio da herança, permitindo que a chamada dos membros citados anteriormente resulte na execução de ações diferentes.
   - **EX:**
     - O mamífero se movimenta.
     - O cachorro se movimenta em terra.
     - O golfinho se movimenta em água.

---

6. **(1.0 pontos)** Na Herança, a ordem de chamada dos construtores e destrutores é distinta. Justifique o porquê, apresentando também qual ordem as chamadas ocorrem.
   - **R:** Os construtores, quando chamados, são inicializados começando pela superclasse e seguem pelas derivadas. Isso ocorre por conta das classes derivadas utilizarem aspectos herdados da superclasse. Por conta disso, o destrutor é chamado de forma inversa, para que não sejam causados erros por conta da hierarquia das classes.

---

7. **(1.5 pontos)** Verifique as afirmações abaixo em relação Programação Orientada a Objetos, colocando **V** ou **F**:
   - 1) (**V**) A programação orientada a objetos tem como principais objetivos reduzir a complexidade no desenvolvimento de software e aumentar sua produtividade.
   - 2) (**V**) Em C++, classes podem possuir quantos métodos construtores e destrutores forem necessários.
   - 3) (**F**) Variáveis membro de uma classe com o qualificador *static* são únicas e reservadas nos objetos, não sendo possíveis serem acessadas externamente, mesmo quando qualificadas como públicas.
   - 4) (**V**) Por meio do mecanismo de sobrecarga, vários métodos de uma mesma classe podem ter o mesmo nome, desde que suas listas de parâmetros sejam diferentes. No entanto, devem obrigatoriamente apresentar o mesmo tipo de retorno.
   - 5) (**V**) Na herança, as subclasses tornam-se mais específicas em relação à superclasse, podendo adicionar novos atributos e métodos que serão acessíveis ao longo de toda a estrutura hierárquica.
   - 6) (**F**) No uso de herança entre uma classe base e uma derivada, tanto construtores como destrutores são primeiro executados na classe base e depois na derivada.
   - 7) (**V**) Em uma Herança de 3 níveis hierárquicos: é possível possuir várias instâncias de objetos em qualquer nível. Uma forma de polimorfismo é utilizar um ponteiro do tipo do primeiro nível hierárquico, e utilizá-lo para representar qualquer instância em qualquer nível hierárquico das subclasses, exceto uma instância do primeiro nível.
   - 8) (**V**) No paradigma orientado à objetos a comunicação entre objetos se dá através de troca de mensagens.
   - 9) (**F**) Herança múltipla é descrita na orientação à objetos como um mecanismo onde uma classe derivada herda a partir de duas classes bases, e possui características de ambas. Porém, nenhuma Linguagem de Programação implementa esse mecanismo devido ao auto nível de ambiguidade.