Universidade Federal de Itajubá - Prova Bimestral de ECOP03
Programação Orientada a Objetos

Nome:		**Nº**	

Duração aprox. 1:50 h. Data de entrega - 17/10/22 - 10:10h até 22:10

Instruções:

- Prova individual.
- Questões dissertativas pessoais.
- A interpretação faz parte da prova.
- Responder na própria prova, inserindo textos (cor vermelha) e fotos/print screen, se necessário.
- Salvar o arquivo em PDF com o nome: prova1-NomeCompleto-NumeroMatricula.pdf
- Enviar o arquivo na tarefa aberta no SIGAA.

1ª Questão: Para a classe CVetor declarada abaixo:

```
//arquivo vetor.h
//header file para classe vetor
#ifndef VETOR_H
#define VETOR_H

const int inicio=0;

class CVetor{

private:
      double* m_vet;          //este e' o vetor
      int   m_tam;            //tamanho máximo do vetor,
public:
      CVetor (int tam);       //construtor, aloca memória para o vetor.
      void Atribui(int index, double valor); //altera uma posição do vetor
      double Conteudo(int index);         //retorna conteúdo de posição do vetor
      double Maximo(void);                //retorna o maior elemento do vetor
      double Primeiro(void);              //primeiro índice do vetor
      double Ultimo(void);                //último índice do vetor
      ~CVetor() {delete [] m_vet;}        //destrutor inline
};
#endif
```

a) (4pts) Sobrecarregue o operador de leitura (>>). Declaração e implementação

```
friend ostream& operator << (ostream& saida, CVetor CV);  // Resposta do
item (a) da questao 01;

ostream& operator << (ostream& saida, CVetor CV)  // Resposta do item (a)
da questao 01
{
      for (int i = 0; i < CV.m_tam; i++)
      {
            saida << i << CV.m_vet[i] << "\t";
      }
      return saida;
}
```

b) (4pts) Sobrecarregue o operador de impressão (<<). Declaração e implementação

```
44  friend istream& operator >> (istream& entrada, CVetor CV); // Resposta do
    item (b) da questao 01;
45
46  istream& operator >> (istream& entrada, CVetor CV) // Resposta do item (b)
    da questao 01
47  {
48      cout << "Insira o tamanho do vetor: ";
49      cin >> CV.m_tam;
50      int j = 1;
51
52      for (int i = 0; i < CV.m_tam; i++)
53      {
54          cout << "Digite o " << j << " valor: ";
55          entrada >> CV.m_vet[i];
56          j += 1;
57      }
58      return entrada;
59  }
```

c) (4pts) Sobrecarregue o operador de acesso (**[ ]**), de modo que ele possa ser usado nos dois lados de uma atribuição.

```
64  double& operator [] (int); // Resposta do item (c) da questao 01;
65
66  double& CVetor :: operator [] (int aux) // Resposta do item (c) da questao
    01;
67  {
68      if (aux >= 0 && aux < m_tam)
69      {
70          return m_vet[aux];
71      }
72      else
73      {
74          return m_vet[0];
75      }
76  }
77
```

d) (4pts) Para que a classe funcione corretamente sem bugs, pelo menos dois métodos devem ser acrescentados a ela. Quais são eles, e por que eles devem ser implementados? Implemente pelo menos um deles.

É necessário que haja mais dois métodos: O método de cópia e método de atribuição. O primeiro fara com que o vetor não perca sua referência e o segundo armazena os dados que o usuário coloque no vetor.

```
78    CVetor(const CVetor&); // Resposta do item (d) da questao 01;
79    CVetor operator = (const CVetor CV); // Resposta do item (d) da questao 01;
80
81    CVetor :: CVetor(const CVetor& CV) // Resposta do item (d) da questao 01;
82    {
83          m_tam = CV.m_tam;
84          m_vet= new double[m_tam];
85          for (int i = 0; i < CV.m_tam; i++)
86          {
87                m_vet[i] = CV.m_vet[i];
88          }
89    }
90
```

e) (4pts) Escreva um programa que utilize todos os membros dessa classe.

Link: https://github.com/luziscarine/POO_Programacao_Orientada_a_Objeto/tree/main/teorico/prova

2ª Questão. Responda as seguintes questões teóricas:

a) (5 pts) Cite pelo menos uma restrição que deve ser observada para a utilização de vetores de objetos.

Uma restrição é a obrigação do usuário em criar um método de cópia para um atribuído de um vetor.

b) (5 pts) O que significa criar um objeto?

Criar um objeto é declarar, na Função Principal, um novo tipo criado pelo usuário tendo como base o desenvolvimento de uma classe.

c) (5 pts) Quais os componentes de uma Descrição de Classe?

Atributos: normalmente declarados no privado ou protegido da classe;
Métodos: normalmente declarados no público de uma classe;

d) (5 pts) Explique o conceito de encapsulamento?

O encapsulamento é o conceito que define a privacidade dos atributos em uma classe a fim de não tornar os dados essenciais para o funcionamento do programa editáveis para terceiros.

e) (5 pts) Para que serve um Construtor de Cópia? Qual sintaxe utilizada para declará-lo?

O construtor de cópia é necessário quando deseja-se criar um método para acessar os dados de um vetor sem perder a referência (dado inserido inicialmente pelo usuário) deste vetor.

Declaração: → Classe (const Classe&)

f) (5 pts) Como o compilador diferencia entre os operadores unários de incremento e pós-incremento, ao sobrecarregarmos?

Devemos utilizar o parâmetro (int) de forma a deixai-lo pós-fixado no operador;

g) (5 pts) Existe alguma maneira de se acessar um membro private de uma determinada classe fora de seu escopo? Se houver, dê um exemplo em código.

Sim, há a possibilidade de acessar dados privados a partir da função friend:

3ª Questão (5 pts).Preencha as lacunas de cada uma das sentenças a seguir, utilizando as palavras abaixo:

~~atributos~~ – ~~main~~ – ~~classes~~ – ~~herança~~ – ~~encapsulamento~~

a) Os objetos permitem a prática do Encapsulamento — embora eles possam se comunicar entre si por meio de interfaces bem definidas, normalmente não têm autorização para descobrir como outros objetos são implementados.
b) Os programadores C++ concentram-se na criação de Classes, que contêm campos e o conjunto de métodos que manipulam esses campos, além de fornecer serviços para clientes.
c) Uma nova classe de objetos pode ser convenientemente criada por Herança — a nova classe (chamada subclasse) começa com as características de uma classe existente (chamada superclasse), personalizando-as e talvez adicionando características próprias.
d) O tamanho, forma, cor e peso de um objeto são considerados Atributos da classe dele.
e) Aplicativos C++ iniciam a execução no método Main.

4ª Questão. (10pts) Como o Compilador interpreta as seguintes expressões sabendo que todos os operadores necessários estão sobrecarregados como funções membros da classe que modelou os objetos A, B, C, D e E.

a) A+B*C/(D+E)

```
// Primeiro o compilador executa os dados em parênteses
Primeira operação = (D + E);

// O compilador seguira a ordem de precedência na aritmética da esquerda para a direita fara primeiro os produtos e as razoes e por fim fara a soma:

Segunda operação = B * C;
Terceira Operação = (B * C) / (D+E)
Quarta Operação = A + (B * C) / (D + E)
```

b) A--

```
Valor A; // Valor declarado

A--:  A – 1;

Cout <<  A << endl;

// Logo se A fosse 10 o compilador retornaria 9;
```

c) (float) (A*B+C-D*E)

```
 // Primeiro o compilador entenderá que o numero a ser calculado é do tipo float
// Depois o compilador fará as seguintes operações:
/*
        A* B;   -D* E;   A*B + C – D * E;
```

5ª Questão. (15pts) Diga se a frase é verdadeira ou falsa. Caso seja falsa destacar o trecho que a torna falsa.

(V) Em C++, o programador pode definir seus próprios tipos de dados, ao invés de somente utilizar os tipos nativos como int e float.

(F) Em C++ quando se altera um atributo `static` de uma classe, esse valor fica disponível somente para o objeto que realizou essa alteração.

(V) O construtor de cópia é utilizado na passagem de parâmetros de objetos por valor, do tipo da classe que o implementa.

(V) Para uma classe denominada CNomeClasse, a sintaxe para declarar um construtor de cópia é: `CNomeClasse ( const CNomeClasse& );`

(V) Sobrecarregar um operador significa definir uma nova funcionalidade para aquele símbolo na linguagem.

(V) O construtor de inicialização serve para copiar um objeto existente para um outro que está sendo criado.

(F) Todo código escrito em C++ precisa, obrigatoriamente, ser colocado dentro de classes.

(F) Considerando uma hierarquia de classes definida em C++, uma função virtual pura não precisa ser implementada na classe base, para que objetos das classes derivadas tenham acesso a ela.

(F) Em C++ é possível sobrecarregar os operadores de atribuição como funções *friend*.

(V) Os componentes de uma descrição de classe são: Declaração de Atributos, Declaração de Métodos e Implementação de Métodos.

(F) `double d2 {2.3};` não é uma inicialização válida para um número real em C++11.

(F) `private` e `public` são especificadores de acesso em C++11.

(V) Funções *friend* são utilizadas para ter acesso privilegiado aos membros de uma classe.

(V) Em orientação a objetos, criar um objeto significa definir uma variável do tipo de uma classe.

(V) Em Orientação a Objetos a Resolução Dinâmica de Métodos permite que o programador crie códigos genéricos que podem ser reaproveitados quando novas classes são inseridas na hierarquia.

6ª Questão (15 pts): Para as classes abaixo responda:

```
class ClasseBase
{
   public:
       int m_A1;           // Declarar membros como public.
   protected:
       int m_A2;           // Declarar membros como protected
       int F1Base(void);
   private:
       int m_A3;           // Declarar membros como private
};
```

```
class Derivada1 : public ClasseBase
{
      private:
      // Declarar membros private
            int m_B1;
      protected:
      // Declarar membros protected
            int m_B2;
            int F1D1(void);
      public:
      // Declarar membros public.
            int m_B3;
            int F2D1(void);
};
```

```
class Derivada2 : private ClasseBase
{
      private:
      // Declarar membros private.
            int m_C1;
      protected:
      // Declarar membros protected
            int m_C2;
            int F1D2(void);
      public:
      // Declarar membros public.
            int m_C3;
            int F2D2(void);
};
```

a) Quais membros que podem ser acessados pelos métodos da classe **Derivada1**. Monte uma tabela com seus nomes e respectivos graus de acesso.

Atributo	Grau de Acesso
m_A1	Publico
m_A2	Protegido
F1Base(void)	Protegido
m_B2	Protegido
m_B3	Publico
F1D1(void)	Protegido
F1D2(void)	Publico
m_B1	privado

b) Quais membros podem ser acessados pelos métodos da classe **Derivada2**. Monte uma tabela com seus nomes e respectivos graus de acesso.

Atributo	Grau de Acesso
m_A1	Publico
m_A2	Protegido
m_C1	privado
m_C2	Protegido
F1D2(void)	Protegido
m_C3	Publico
F2D2(void)	Publico

c) Se a classe **Derivada2** for utilizada como base publica de uma terceira classe, chamada **Derivada3** quais membros de dados poderão ser acessados pelos métodos desta nova classe. Coloque seus respectivos graus de acesso. Como ficaria o acesso para os membros de **ClasseBase** nesta nova classe?

Atributo	Grau de acesso
m_C3	public
F2D2(void)	Public
m_C2	Protegido

F1D2	Protegido