UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 – Linguagem de Programação I

Avaliação II
05/06/2018

ALUNO: _____

Instruções Gerais

Leia atentamente as instruções a seguir antes de iniciar a avaliação:

1. Não esqueça de colocar o seu nome completo nesta folha.
2. Esta folha de questões, devidamente identificada, deve ser entregue ao professor ao final da avaliação.
3. A resolução desta avaliação é **individual**.
4. Você terá duas aulas (totalizando 1h40min) para responder a todas as questões. Por isso, gerencie bem o seu tempo.
5. Esta avaliação vale 3,00 (três) pontos no total, sendo os pontos distribuídos em 1,50 para as questões declaradas como teóricas e 1,50 para as questões declaradas como práticas. O valor de cada questão teórica é fornecido junto ao seu respectivo enunciado, enquanto as questões práticas serão valoradas em seu conjunto.
6. Leia atentamente o enunciado de cada questão antes de iniciar a sua resposta.

Questão 1. (teórica - 0,50) Explique as diferenças entre um método virtual, um método virtual puro, uma classe virtual e uma interface em C++ (ainda que interface não seja um conceito nativo da linguagem). Qual a utilidade de uma interface? Dê um exemplo de interface.

Questão 2. (teórica - 0,50) Explique os conceitos de contentores (*containers*) e iteradores (*iterators*) usado pela STL (*Standard Template Library*) no C++. Explique ainda a razão pela qual os algoritmos implementados na STL se baseiam em iteradores e não em contentores.

Questão 3. (teórica - 0,50) O código a seguir explora significativamente os recursos da STL para executar uma tarefa bastante útil. Analise o código apresentado e descreva a sua função. Descreva em detalhes o uso de recursos da STL. Por fim, escreva (sim, de próprio punho!!) um programa exemplo que demonstre a correta utilização desta função.

```
1 template<typename InputIterator>
2 InputIterator misterio(InputIterator first, InputIterator last) {
3   InputIterator it =
4       std::lower_bound(first, last, std::accumulate(first, last, 0.0)
5           / (double) std::distance(first, last));
6   return it;
7 }
```

Questão 4. (prática) O programa a seguir apresenta alguns problemas de gerenciamento de memória, que acabam por causar o erro de "falha de segmentação"(*segmentation fault*). Compile e execute o programa para verificar por sí mesmo. Em seguida, analise o código, utilizando as ferramentas GDB e Valgrind, a fim de identificar os problemas. Aponte cada problema encontrado, proponha a solução adequada e demonstre que o problema foi solucionado.

```
1
2  // Esse programa esta mesmo "bugado"!!!
3
4  #include<iostream>
5
6  using namespace std;
7
8  int main(int argc, char const *argv[])
9  {
10
11   double* d = new double;
12
13   for(unsigned int i = 0; i < 3; i++) {
14     d[i] = 1.5 + i;
15   }
16
17   for(unsigned int i = 2; i >= 0; i--) {
18     cout << d[i] << endl;
19   }
20
21 }
```

Questão 5. (prática) Considere um programa em C++ que leia um tempo no formato HH:MM:SS e imprima o total em segundos, usando uma classe Tempo. Mais uma vez, Teobaldo já iniciou o código, mas precisa de sua ajuda para completar.

```
1  #include <iostream>
2  #include <iomanip>
3
4  using namespace std;
5
6  class Tempo
7  {
8      private:
9          int hh,mm,ss;
10     public:
11         // Le os dados do tempo a partir da entrada padrao
12         void lerTempo(void);
13         // Retorna o tempo em segundos
14         int converteEmSegundos(void);
15         // Imprime o tempo no formato HH:MM:SS e o seu total
16         // em segundos
17         void mostraTempo(void);
18 };
19
20 // Implementar os metodos...
21
22 int main()
23 {
24     Tempo T;
25
```

```
26      T.lerTempo();
27      T.mostraTempo();
28
29      return 0;
30  }
```

Questão 6. (prática) Mostre ao Teobaldo o que você aprendeu sobre sobrecarga de operadores e altere o código do programa da questão anterior para substituir os métodos lerTempo() e mostraTempo() pelos operadores de extração e inserção, respectivamente.

Questão 7. (prática) Teobaldo está estudando o mecanismo de herança em C++, mas está com dificuldades em codificar a seguinte hierarquia de classes. Ele sabe que a função main() está correta, mas que há problemas com as definições de herança.

Aponte os problemas de uso de herança no código do Teobaldo, descrevendo claramente o problema e proponha as correções necessárias para que o programa funcione corretamente.

Indique e justifique a ordem de construção e destruição de objetos das classes Aluno e Gerente.

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Pessoa {
6  public:
7    string m_nome;
8    int m_idade;
9  public:
10   Pessoa(string nome_, int idade_): m_nome(nome_){};
11   ~Pessoa(){};
12   string getNome(){ return m_nome; };
13   int getIdade(){ return m_idade; };
14 };
15
16 class Empregado : private Pessoa {
17 private:
18   string m_matricula;
19   double m_salario;
20 public:
21   Empregado(string nome_, int idade_, string matricula_, double
        salario_):
22     Pessoa(nome_, idade_), m_matricula(matricula_), m_salario(salario_
         ){};
23   ~Empregado(){};
24   string getMatricula(){ return m_matricula; };
25   double getSalario(){ return m_salario; };
26 };
27
28 class Aluno : public Pessoa {
29 private:
30   string m_matricula;
31 public:
```

```
32    Aluno(string nome_, int idade_, string matricula_):
33      Pessoa(nome_, idade_), m_matricula(matricula_){};
34    ~Aluno(){};
35    string getMatricula(){ return m_matricula; };
36  };
37
38  class Vendedor : private Empregado {
39  private:
40    double m_meta_mensal;
41    double m_desconto_nivel1;
42  public:
43    Vendedor(string nome_, int idade_, string matricula_, double
          salario_, double meta_, double desconto_):
44      Empregado(nome_, idade_, matricula_, salario_),
45      m_meta_mensal(meta_), m_desconto_nivel1(desconto_){};
46    ~Vendedor(){};
47    double getMetaMensal(){ return m_meta_mensal; };
48    double getDescontoN1(){ return m_desconto_nivel1; };
49  };
50
51  class Gerente : private Empregado {
52  private:
53    string m_setor;
54    double m_desconto_nivel2;
55  public:
56    Gerente(string nome_, int idade_, string matricula_, double salario_
          , string setor_, double desconto_):
57      Empregado(nome_, idade_, matricula_, salario_),
58      m_setor(setor_), m_desconto_nivel2(desconto_){};
59    ~Gerente(){};
60    string getSetor(){ return m_setor; };
61    double getDescontoN2(){ return m_desconto_nivel2; };
62    void imprimeDados(){ cout << "Nome: " << m_nome  << "\t"
63              << "Idade" << m_idade << "\t"
64              << "Matricula: " << getMatricula() << endl
65              << "Salario: " << getSalario() << "\t"
66              << "Setor: " << m_setor << endl
67              << "Desconto N2: " << m_desconto_nivel2
68              << endl; };
69  };
70
71  int main(int argc, char const *argv[])
72  {
73    Aluno a("Maria de Lourdes", 22, "98765432-1");
74    Gerente g("Emiliano Emilio", 45, "666000666-1", 5780.00, "
          Departamento Financeiro", 7.5);
75    g.imprimeDados();
76    return 0;
77  }
```

Questão 8. Discuta as implicações de definirmos o seguinte template na classe ou programa principal

em C++.

```
template <typename T>
ostream& operator<<(ostream& ostr, const T &x)
{
    x.print(ostr);
    return ostr;
}
```

Questão 9. (prática) Sua vez de mostrar que você entende de ponteiros inteligentes. Após compilar e executar o progama a seguir, comente o código (completando as marcações indicadas C1..C9), indicando o efeito de cada operação que segue o comentário. Em cada comentário, indique se há alteração na saída padrão e qual a razão. Para facilitar, os comentários já foram iniciados.

```
#include <iostream>
#include <memory>

using namespace std;

class Pessoa
{
private:
   string m_nome;
public:
   Pessoa(string nome_):m_nome(nome_){};
   ~Pessoa(){ cout << "Destruindo " << m_nome << "... " << endl; };
   string getNome(){ return m_nome; };
};

void quemEhEstaPessoa(Pessoa * p){
   cout << "Esta pessoa eh: " << p->getNome() << endl;
}

int main(int argc, char const *argv[])
{
   // C1: Cria um ponteiro inteligente ptr1 que aponta unicamente
   //     para a instancia (de Pessoa) "Maria". Nao altera a saida.
   unique_ptr<Pessoa> ptr1(new Pessoa("Maria"));

   // C2: Utiliza a notacao de ponteiro, permitida pelo ponteiro
   //     inteligente, para invocar o metodo getNome(). Imprime
   //     o string "Maria" na saida padrao, como resultado da
   //     execucao do metodo getNome().
   cout << ptr1->getNome() << endl;

   // C3:
   cout << (*ptr1).getNome() << endl;

   // C4:
   quemEhEstaPessoa(ptr1.get());

   // C5:
```

```
39      unique_ptr<Pessoa> ptr2(std::move(ptr1));
40
41      // C6:
42      ptr1.reset(new Pessoa("Joao"));
43
44      // C7:
45      ptr2.reset(new Pessoa("Marcos"));
46
47      // C8:
48      ptr1.reset();
49
50      // C9:
51
52      return 0;
53  }
```

Questão 10. (prática) Agora que você já demonstrou que entende de ponteiros inteligentes. Após compilar e executar o progama a seguir, comente o código (completando as marcações indicadas C1..C6), indicando o efeito de cada operação que segue o comentário. Em cada comentário, indique se há alteração na saída padrão e qual a razão. Desta vez você certamente consegue realizar o trabalho sozinho.

```
1   #include <iostream>
2   #include <memory>
3
4   using namespace std;
5
6   class Pessoa
7   {
8   private:
9       string m_nome;
10  public:
11      Pessoa(string nome_):m_nome(nome_){};
12      ~Pessoa(){ cout << "Destruindo " << m_nome << "... " << endl; };
13      string getNome(){ return m_nome; };
14  };
15
16  void quemEhEstaPessoa(Pessoa * p){
17      cout << "Esta pessoa eh: " << p->getNome() << endl;
18  }
19
20  int main(int argc, char const *argv[])
21  {
22      // C1:
23    auto ptr = make_shared<Pessoa>("Sergio");
24
25      // C2:
26    cout << ptr->getNome() << endl;
27
28      // C3:
29    quemEhEstaPessoa(ptr.get());
```

```
30
31      // C4:
32    shared_ptr<Pessoa> aquelaPessoa = ptr;
33
34      // C5:
35    ptr.reset(new Pessoa("Thiago"));
36
37      // C6:
38    aquelaPessoa.reset();
39
40    return 0;
41  }
```