

Lista de Exercícios 01

01/05/2018

Questão 1. 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 si 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 2. 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 3. 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 4. 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.

```
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
44         salario_, double meta_, double desconto_):
45         Empregado(nome_, idade_, matricula_, salario_),
46         m_meta_mensal(meta_), m_desconto_nivel1(desconto_){};
47     ~Vendedor(){};
48     double getMetaMensal(){ return m_meta_mensal; };
49     double getDescontoN1(){ return m_desconto_nivel1; };
50 };
51
52 class Gerente : private Empregado {
53 private:
54     string m_setor;
55     double m_desconto_nivel2;
56 public:
57     Gerente(string nome_, int idade_, string matricula_, double salario_,
58         string setor_, double desconto_):
59         Empregado(nome_, idade_, matricula_, salario_),
60         m_setor(setor_), m_desconto_nivel2(desconto_){};
61     ~Gerente(){};
62     string getSetor(){ return m_setor; };
63     double getDescontoN2(){ return m_desconto_nivel2; };
64     void imprimeDados(){ cout << "Nome: " << m_nome << "\t"
65         << "Idade" << m_idade << "\t"
66         << "Matricula: " << getMatricula() << endl
67         << "Salario: " << getSalario() << "\t"
68         << "Setor: " << m_setor << endl
69         << "Desconto N2: " << m_desconto_nivel2
70         << endl; };
71 };
72
73 int main(int argc, char const *argv[])
74 {
```

```
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 }
```

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

Questão 5. Discuta as implicações de definirmos o seguinte template na classe ou programa principal em C++.

```
1  template <typename T>
2  ostream& operator<<(ostream& ostr, const T &x)
3  {
4      x.print(ostr);
5      return ostr;
6  }
```

Questão 6. Sua vez de mostrar que você entende de ponteiros inteligentes. Após compilar e executar o programa 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.

```
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: Cria um ponteiro inteligente ptr1 que aponta unicamente
23     //      para a instancia (de Pessoa) "Maria". N o altera a sa da .
24     unique_ptr<Pessoa> ptr1(new Pessoa("Maria"));
25
26     // C2: Utiliza a notacao de ponteiro, permitida pelo ponteiro
27     //      inteligente, para invocar o metodo getNome(). Imprime
28     //      o string "Maria" na sa da padr o, como resultado da
```

```
29 //      execu o do m todo getNome().
30 cout << ptr1->getNome() << endl;
31
32 // C3:
33 cout << (*ptr1).getNome() << endl;
34
35 // C4:
36 quemEhEstaPessoa(ptr1.get());
37
38 // 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 7. Agora que você já demonstrou que entende de ponteiros inteligentes. Após compilar e executar o programa 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 }
```