Universidade Federal de Minas Gerais
departamento de ciência da computação
Disciplina: Algoritmos e Estrutura de Dados II (DCC004)

Professor: Renato Martins (renato.martins AT dcc.ufmg.br)
https://www.dcc.ufmg.br/~renato.martins/courses/DCC004
2° Semestre de 2018

Lista Revisão – Armazenamento de Dados na Memória, TADs e Programação Orientada a Objetos

1. Quais as vantagens e desvantagens de alocação de memória no `heap` e na pilha (`stack`)? Qual a política de alocação da `stack` (FIFO ou LIFO)?

2. Em que região da memória (stack ou heap) as seguintes variáveis serão alocadas?

   (a) `int *p = new int[10]`

   (b) `std::string s[20]`

   (c) `Image img;`, em que
   ```
   struct Image{
   int w,h;
   uint8_t *data;};
   ```

3. Quais as diferenças entre ponteiro e referência em C++?

4. Qual a diferença entre `delete` e `delete[]`?

5. Qual estrutura de dados da STL é indicada para guardar um número indefinido de itens preservando a ordem de inserção? (vector, set ou map)?

6. Implemente o tipo abstrato de dados (TAD) matriz de números reais: contendo número de linhas (int), colunas (int) e os dados do tipo float. O seu TAD deve suportar as seguintes operações:

   (a) **criar_matriz**: alocar dinamicamente uma matriz.

   (b) **apagar_matriz**: desalocar a matriz.

   (c) **adicionar_elemento**: adição de um elemento a uma posição específica da matriz.

   (d) **imprimir**: imprime todos os elementos da matriz.

   Escreva um pequeno programa para testar o seu TAD.

7. Implemente um TAD **músico** em C++:

   (a) Cada músico possui os atributos `nome`, `tipo de instrumento` e `lista de apresentações`.

   (b) Implemente as operações: `Musico`: que inicializa os dados do músico. `imprime`: mostra todos os dados do músico.

Escreva um pequeno programa para testar o seu TAD.

8. Implemente um TAD **estacionamento de veículos** em C++. O estacionamento é um conjunto que pode conter veículos do tipo `caminhão, ônibus, carro, moto` e `bicicleta`. O numero de vagas que um caminhão ou ônibus ocupam é de 10 vagas. O carro ocupa uma vaga. Em uma vaga cabem 2 motos e em uma vaga cabem 10 bicicletas.

(a) O seu estacionamento tem uma capacidade limitada de vagas (alocada dinamicamente) e contém uma lista para cada um dos tipos de veículos que estão no estacionamento.

(b) Implemente as operações: `Estacionamento`: que inicializa os dados do estacionamento. `imprime`: mostra o número de veículos por modalidade no estacionamento. `verificar_vaga`: checa o número de vagas no estacionamento. `insere_veiculo`: insere um veículo no estacionamento. `mostra_vagas`: imprime número de vagas livres e a ocupação do estacionamento.

9. Qual a diferença entre classe e objeto?

10. Os componentes de uma classe podem ser membros de instância ou membros de classe (estáticos). Qual a diferença?

11. O que significa usar `this` dentro de uma classe?

12. Comente os seguintes princípios fundamentais de programação orientada a objetos: abstração, encapsulamento, herança e polimorfismo.

13. De que modo o polimorfismo permite programar em um nível "geral" ao invés de programar em um nível "específico"? Discutir as vantagens da programação no nível "geral".

14. Descreva com exemplos a visibilidade de atributos com os modificadores de acesso `private`, `protected` e `public`. Qual a visibilidade padrão de métodos e atributos em uma classe em C++?

15. Em que situações seria recomendado o uso de um construtor de uma classe como `protected`? E como `private`? Dê um exemplo ilustrativo para cada.

16. Quais são as utilidades de funções virtuais puras (abstratas) em uma classe?

17. Qual a saída do seguinte programa?

```
1  class A {
2  public:
3  void f() {
4  std::cout<<"Metodo f de A"<<std::endl;
5  }
6  };
7  class B: public A {
8  public:
9  void f() {
10 std::cout<<"Metodo f de B"<<std::endl;
11 }
12 };
13
14 int main() {
15 A *a = new A();
16 A *ab = new B();
17 B* b = new B();
```

```
18  a->f();
19  ab->f();
20  b->f();
21  return 0;
22  }
```

Qual será a saída desse programa caso o método na classe A for definido como `virtual void f() { std::cout<<"Metodo f de A<<std::endl; }` ?

18. Quais as diferenças de uma classe concreta, abstrata e interface?

19. Crie uma estrutura hierárquica que contenha as classes **Animal** (classe abstrata), **Gato** e **Marmota**. A classe **Animal** possui o atributo _nome e os métodos são todos abstratos e possuem a seguinte assinatura:

(a) protected: std::string _nome

(b) public: std::string get_nome()

(c) public: void dormir(int horas)

Estes métodos são implementados nas subclasses **Gato** e **Marmota**. Mostre o seu diagrama UML da hierarquia das classes.

20. Considere uma classe `Pilha`. Implemente a classe e suas duas principais operações, `push` e `pop`. `Push` é um método que deve lançar uma exceção do tipo `PilhaCheia` sempre que não couber mais elementos na pilha. `Pop` deve lançar uma exceção do tipo `PilhaVazia` sempre caso a pilha não tenha mais elementos a serem retirados. Ambas exceções devem herdar da classe `PilhaExcecao`. Implemente as três classes de exceção e os métodos que as lançam (`push e pop`). Implemente também uma classe `Teste` que mostra como criar uma pilha e invocar os métodos `push` e `pop` com os respectivos tratamentos de exceções.

21. Crie a seguinte hierarquia de classes:

(a) Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;

(b) Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;

(c) Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;

(d) Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.

No programa principal, pergunte ao usuário quantas formas ele deseja criar. Em seguida, para cada forma, pergunte se deseja criar um quadrado, um retângulo ou um círculo, solicitando os dados necessários para criar a forma. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os dados (lados ou raio); (b) os perímetros; e (c) as áreas de todas as formas. Para (b) e (c), tire vantagem do polimorfismo, enquanto que para (a) utilize typeid e downcast.