Lista 3

Aluno: Nathann Zini dos Reis

Matrícula: 19.2.4007

Main1.cpp

Q.1 - a)

b) A maneira como o código está disposto/indentado dificulta o entendimento do mesmo. Além de os nomes das variáveis não serem intuitiva para entender o que elas representam.

Q.2-

```
class Casa {
    float orc;
    int a;
public:
    Casa( float o ) : orc(o){
        cout << "Casa criada..." << endl;
    }
    void setOrc( float o ) {
        orc = o;
    }
```

```
    float getOrc( void ) {
        return orc;
    }
};
```

Q.3 - O código cria uma variável do tipo Casa e em seguida dois ponteiros que apontam para o endereço de memória da primeira variável do tipo Casa criada anteriormente. Ou seja, cria a variável C1, e em seguida ponteiro C2 que aponta para o endereço de memória de C1 e outro endereço de memória C3 que recebe o endereço de C1.

Nas impressões, imprime o valor que foi iniciado C1, e em seguida é atribuido outro valor a C1, porém, como C2 e C3 apontam para o endereço de C1, ao imprimir, imprime o novo valor atribuido a C1. Em seguida imprime o endereço de C2 e C3 que é o mesmo e é o mesmo que o de C1.

Q.4 - O construtor foi chamado apenas uma vez, pois os demais são ponteiros que apontam para a variável que foi criado anteriormente chamando o construtor Casa.

Main2.cpp

Q.1 - a) A declaração do operador como sendo Friend é necessária para que os dados privados do objeto casa possam ser acessados por ele.

b) O operador "<<" agora recebe o objeto Casa e imprime o valor de orc referente àquele objeto

Q.2 - É criado um objeto do tipo Casa e um ponteiro do tipo Casa que aponta pro objeto que foi criado anteriormente. Em seguida é impresso os valores de Orc de cada uma das variáveis e como elas apontam para o mesmo objeto o valor impresso é o mesmo. No primeiro caso é usada a função get para obter os valores e na segunda linha é usado a sobrecarga do operador "<<"

Main3.cpp

Q.1- a) Foi declarada como static para o valor ser mantido ao encerrar a função. Assim, toda vez que a função for chamada novamente ela terá o valor de cont mantido das últimas utilizações.

b) Pois seria necessário inicializar antes de usar para reservar uma porção da memória para ele.

c) Por causa do caractere referente ao fim da string "\0"

d) Não é recomendado a utilização da variável global pois ela permanecerá alocada na memória mesmo quando não estiver mais em uso e/ou quando não for mais ser usada e também porque aquele nome de variável será reservado para ela e impossibilitado de ser usado para outra.

e) Pela necessidade de ser alocada na memória previamente.

Q.2 - Pois não tem destrutor criado, ou seja, haverá acumulo de variável na memória devido à perca de referência

Q.3 - Como o tipo cliente não é um tipo nativo da linguagem, ele dá o erro ao tentar usar o operador "=" para fazer atribuições entre o tipo Cliente. É necessário então fazer a sobrecarga do operador, ou então a memória alocada para Cli2 será perdida e aumentara o consumo da memória.

Depois o programa imprime um texto com o nome da Dra. Beltrana e do Dr. Fulano.

Então, faz um casting para void* imprimindo o endereço de memória.

Q.5 - Como o método é public ele pode ser chamado diretamente por meio do nome da classe, e como é static não é necessário criar um objeto daquela classe para executar aquele código dentro do mesmo arquivo.

Q.6 - O método é declarado como virtual pois ele não tem a implementação na classe mãe, mas nas classes que herdam dela, pois cada uma, provavelmente, terá comportamento distinto para a mesma função.

Q.7 - a) Não é possível instanciar um objeto de uma classe abstrata. É necessário que outra classe herde dela e seja devidamente implementada para então poder instanciar.

b) É declarada como pure virtual para que a implementação seja feita individualmente em cada classe que herda dessa classe abstrata, pois cada uma pode ter um comportamento distinto.

c)
const serve para impedir a troca do conteudo da string, o segundo serve para impedir a troca de ponteiro.

Q.8 -
Como getClasse inicialmente não era um metodo constante, ele só poderia ser chamado por objetos não constantes. Para resolver este problema, basta definir o método como constante (adicionando const ao final da declaração da função), permitindo assim ser chamado por qualquer tipo de objeto.