Nome: Everton Aparecido Ferreira Número: 1201400182

TEORIA (Tópico II)

**1ª. questão –** Cite algumas vantagens das linguagens Orientadas a Objeto sobre as linguagens estruturadas para a construção e manutenção de sistemas de informação.

Uma linguagem orientada a objetos é uma linguagem de programação contendo os dispositivos necessários para a definição das classes de objetos, criação de instâncias desses objetos e regulação da troca de mensagens entre eles.   
 Vantagens:

**Encapsulamento**: cada objeto reúne em si todo o necessário para a implementação de uma funcionalidade, de modo que, ao ser necessária a modificação de uma funcionalidade qualquer não é necessário procurar em todo o programa pelos locais onde as mudanças são necessárias, basta alterar o objeto que cuida desta funcionalidade.  
**Reusabilidade**: o conceito de objeto facilita a reutilização de pedaços de um programa na hora de construir-se outros. Basta saber o que o objeto faz e como comunicar-se com ele.   
**Escalabilidade**: se mais de uma pessoa for participar da criação de um sistema, fica natural a divisão do trabalho, cada uma cuidando de um ou mais objetos específicos, precisando entrar em acordo apenas quanto a respeitar a forma de comunicação entre os objetos.   
**Herança**: os objetos não precisam serem criados todos a partir do nada. Um objeto pode "herdar" o comportamento de outro, alterando apenas uma ou outra coisa necessária. Assim, você poderia criar um objeto "Pessoa" para guardar nome, endereço, telefone, etc. e depois criar a partir dele, por herança, objetos como "Cliente" e "Fornecedor", mudando nestes apenas aquilo que os diferencia de uma "Pessoa" genérica.

**2ª. questão –** Mostrar porque a técnica de “encapsulamento de dados” permite construir projetos de sistemas de informação de maior qualidade.

Porque é capaz de combinar diferentes formas de medir resultados obtidos em diferentes fases do processo em que diferentes aspectos são mais importantes. A Técnica de Encapsulamento de Dados permite combinar indicadores gerando padrões de desempenho a partir dos próprios desempenhos observados.

A técnica de encapsulamento de dados protege os objetos de terem seus atributos corrompidos por outros objetos, escondendo seus detalhes de implantação, uma classe evita que outras fiquem dependentes de sua estrutura interna, protegendo seus dados de acesso indevido.

**3ª. questão –** Porque o “ocultamento de informação” auxilia a produzir projetos mais claros, simples e fáceis de ser entendidos?

Porque se um objeto tem uma interface pública, os outros objetos podem estabelecer comunicação com ele. Mas o objeto mantes informações de métodos privados que podem ser alterados a qualquer momento sem afetar os objetos que dependem dele. Assim não é necessário saber como o objeto é implementado para poder utiliza-lo.

**4ª. questão –** A definição de um método em um arquivo deve vir antes ou depois da declaração do mesmo dentro da classe? Por quê?

A definição vem depois da declaração do método, sendo feita fora da classe.

**5ª. questão –** Porque a inicialização de variáveis privadas de uma classe deve ser feita através do método “construtor”?

Porque o construtor inicializa as variáveis privadas para mais de uma instância da classe.

PRÁTICA (Tópico I)

**1ª. questão –** O objetivo deste projeto é fazer com que o aluno sinta “na prática” quando ocorre a chamada dos métodos “construtor” e “destrutor”. Para isto, crie um “projeto novo” denominado “ConstrutorDestrutorEmAcao.vcxproj” e seu arquivo principal “Principal.cpp”.

Crie uma classe genérica denominada “Arvore”. Crie as variáveis privadas “Nome”, “Idade”, “Altura” com o tipo que você acha que estas variáveis devem ter, sendo que a variável Nome se refere ao nome real de uma árvore como Pereira, Ipê etc. A variável Idade se refere ao atributo idade da árvore real em anos, exemplo: 2,6 anos, 1,5 anos etc. A variável Altura se refere à altura real da árvore exemplo 1,85 m, 10 m etc. Crie o construtor e o destrutor desta classe inicializando estas variáveis com valores a sua escolha. Além de inicializar as variáveis, dentro do próprio corpo do método construtor crie uma instrução para escrever na tela a seguinte frase: “Passando pelo construtor de Arvore”.

No programa principal:

1. Instancie dois objetos do tipo “Arvore” com o nome a1 e a2.
2. Encerrar a execução do programa através da instrução “return 0”.

Compilar e executar o programa observando sua saída. Escrever na caixa a seguir a saída do programa.

Passando pelo construtor de Arvore

Passando pelo construtor de Arvore

Responda:

1. Quem chamou o construtor?

Os Objetos, pois quando você instancia um objeto, automaticamente o método construtor é chamado.

1. Esta chamada foi implícita (invisível para o programador) ou explícita, isto é, dá para ver no código fonte a chamada do construtor?

A chamada é Implícita**.**

1. Em que linha do programa o construtor é chamado?

Na linha que tem Arvore a1, a2;

Repetir o experimento para o destrutor colocando a frase “Passando pelo destrutor da Arvore” no corpo do destrutor. Colocar no espaço a seguir a saída deste novo programa.

Passando pelo construtor de Arvore

Passando pelo construtor de Arvore

Passando pelo destrutor da Arvore

Passando pelo destrutor da Arvore

Responda:

1. Quem chamou o destrutor?

0}

Final do main, quando finaliza o lugar onde foi chamado.

1. Esta chamada foi implícita (invisível para o programador) ou explícita, isto é, dá para ver no código fonte a chamada do destrutor?

Implícita.

1. Em que linha do programa o destrutor é chamado?

Final do main.

**ELEMENTOS DA PROGRAMAÇÃO NECESSÁRIOS PARA ESTE PROJETO**

Para imprimir algo na tela, usa-se um “objeto pronto” denominado “cout”. A declaração de “cout” é feita dentro do arquivo de cabeçalho “iostream.h” o qual já se encontra incluído no arquivo do programa. A sintaxe de uso é a seguinte:

**cout << “Esta frase sera impressa na tela.” << endl;**

Onde,

a) “endl” é uma instrução do C++ que manda o cursor pular uma linha após o

ponto final da frase.

b) “<<” é um operador do C++ que escreve na tela a variável ou constante que

estiver à direita do mesmo.

c) “;” é o símbolo do alfabeto do C++ que significa “fim de uma sentença” em

C++. É obrigatório.

Pode-se, justapor vários operadores “<<” para um mesmo “cout”. Por exemplo, a frase anterior poderia ser escrita da seguinte maneira que o efeito seria o mesmo:

cout << “Esta “ << “frase “ << “sera “ << “impressa “ << “na “ << “tela.” << endl;

Para imprimir o conteúdo de uma variável de qualquer tipo, por exemplo, as variáveis “Nome”, “Idade” e “Altura”:

cout << “Nome = “ << Nome << endl;

cout << “Idade = “ << Idade << endl;

cout << “Altura = “ << Altura << endl;

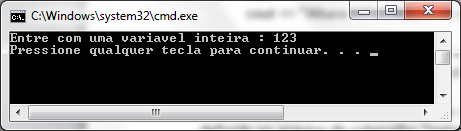
Para ler uma variável digitada no teclado deve-se usar o objeto “cin” também declarado no arquivo de cabeçalho “iostream.h”.

i) Para ler uma variável “int” já definida anteriormente:

cout << “Entre com uma variavel inteira : “;

cin >> NomeDaVariavel;

Ao executar este programa:



E, a variável NomeDaVariavel fica com o valor 123.

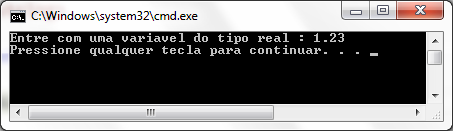
ATENÇÃO: NAS CONSTANTES LITERAIS NÃO USAR ACENTOS DA LINGUA PORTUGUESA COMO EM “VARIAVEL”.

ii) Para ler uma variável “float” já definida anteriormente:

cout << “Entre com uma variavel do tipo real : “;

cin >> NomeDaVariavel;

Ao executar este programa:



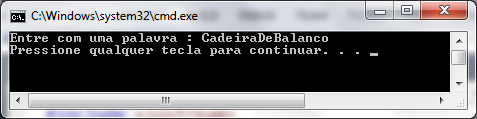
E, a variável NomeDaVariavel fica com o valor 1,23.

iii) Para ler um vetor de caracteres sem espaço entre eles:

cout << “Entre com uma palavra : “;

cin >> NomeDoVetor;

Ao executar este programa:



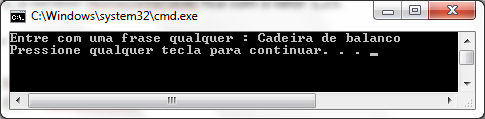
E, a variável NomeDoVetor fica com o valor “CadeiraDeBalanco”.

iv) Para ler um vetor de caracteres com espaço em branco entre eles:

cout << “Entre com uma frase qualquer : “

getline(cin, NomeDaVariavel);

Ao executar este programa:



E, a variável NomeDoVetor fica com o valor “Cadeira de balanco”.

Esta função getline é da biblioteca da linguagem C++ e faz parte da linguagem padrão. Para ela funcionar corretamente é necessário definir a variável cujo nome é “NomeDaVariavel” como sendo do tipo “string”, ou seja, um objeto da classe “string” conforme se mostra a seguir:

string vetor;

cout << "Entre com uma frase qualquer : ";

getline(cin, vetor);

Esta classe também já está declarada no arquivo de cabeçalho “string.h”.