1. Explique o funcionamento do mecanismo chamado “call-back function”.

O mecanismo "callback function" funciona da seguinte maneira: você pode passar funções (callbacks) como parâmetros para personalizar a lógica de execução do seu programa em tempo de execução.

2. Explique o funcionamento do código contido no arquivo “main.cpp”.

Código implementado para fazer manipulações nas janelas do Windows utilizando a biblioteca windows.

3. O que o código contido no arquivo “main1.cpp” faz?

O algoritmo presente no arquivo main1.cpp realiza a conversão de um caractere minúsculo em maiúsculo, sendo o caractere 's' o método de interrupção do algoritmo. Caso o caractere inserido pelo usuário não seja uma letra do alfabeto minúscula, o algoritmo decrementa em 32 o valor da tabela ASCII do caractere inserido.

4. Como este código funciona?

O algoritmo inicia imprimindo "Hello World", declara as variáveis que serão utilizadas. Após isso ele entra em um loop while que se encerra quando o usuário inserir o caractere 's'. O algoritmo lê o input do usuário e o armazena na variável 'c'. Ele calcula a distância entre uma letra do alfabeto maiúscula e uma minúscula na tabela ASCII e armazena isso na variável 'incr'. Logo após ele subtrai o valor de 'incr' do caractere armazenado em 'c' e imprime esse valor na tela.

5. Por que é necessário utilizar as instruções de “type casting”?

O typecasting é necessário para, primeiramente, converter os caracteres em valores inteiros e assim tornar possível a realização das operações matemáticas sobre o valor na tabela ASCII do caractere inserido. Depois, o typecasting será necessário novamente para converter o valor numérico resultante em um caractere.

6. O que significa “0xFF” no código “main2.cpp”?

É o valor 255 em formato de hexadecimal.

7. O que o primeiro loop do código “main2.cpp” faz e como ele funciona?

O primeiro loop no código main2.cpp percorre a string armazenada em 'disciplina' utilizando o ponteiro de char 'pChar'. Antes de entrar no loop, 'pChar' recebe o endereço de memória de 'disciplina'. Dentro do loop é impresso o conteúdo de 'pChar' e seu valor é incrementado em 1 até que o conteúdo dele seja igual a '\0'.

8. O que o segundo loop do código “main2.cpp” faz e como ele funciona?

O segundo loop do código main2.cpp percorre o vetor de inteiros 'algarismos' utilizando o ponteiro 'pInt'. Antes do loop iniciar, 'pInt' recebe o endereço de memória de 'algarismos' e dentro dele é impresso o conteúdo de 'pInt' e o endereço de memória para o qual ele aponta incrementando seu valor

em 1 a cada iteração. O loop para quando o conteúdo de 'pInt' é igual a 9. Depois do loop ele imprime o tamanho de uma variável do tipo int.

9. O que o terceiro loop do código “main2.cpp” faz? Por que ele não funciona?

O terceiro loop do programa, imprime um vetor de caracteres que foi obtido através do “type casting” de um vetor de inteiros para caracteres, porém, esse loop não funciona de maneira adequada. Isso ocorre pois temos uma incompatibilidade entre o tipo do ponteiro e do conteúdo do vetor.

10. O que pode ser concluído sobre a aritmética de ponteiros?

Concluímos que os ponteiros podem ser usados para diversos tipos de variáveis, mas se for inicializado de forma incorreta, o programa não será executado de maneira correta e podem ocorrer erros durante a execução.

11. O código possui duas diretivas de pré-compilação que fazem com que o código “main3.cpp” gere erros. Compile e execute o código com a diretiva ERRO1 ativada e desativada e, então, Bbaseado na saída impressa na tela, explique o que o trecho de código de linha 27 a 32 faz.

Nesse trecho de código, o algoritmo atribui o endereço de memória 0x40300c para a variável 'pInt' e imprime o endereço de memória de 'pInt', o endereço de memória para o qual ele aponta e o conteúdo dentro do endereço de memória para o qual ele aponta. Depois disso há uma tentativa de mudar o valor de pInt, porém ‘66’ não é um valor válido para um endereço de memória. Ele deve ser colocado na forma hexadecimal '0x42'. Por fim, é impresso o endereço de 'variável' e seu conteúdo utilizando type casting para char.

12. O código possui duas diretivas de pré-compilação que fazem com que o código “main3.cpp” gere erros. Compile e execute o código com a diretiva ERRO2 ativada e desativada e, então, baseado na saída impressa na tela, explique o funcionamento do trecho de código entre as linhas 37 e 57.

Nesse trecho de código temos duas partes que realizam as mesmas instruções, com exceção da primeira que tem uma instrução a mais. Primeiramente, a variável 'pInt' é inicializada com NULL, e então, ocorre a impressão de se o ‘ERROR2’ está definido, o endereço da variável 'pInt' e o ponteiro para o qual ela aponta. É verificado também se 'pInt' possui algum conteúdo e logo após é verificado se ele é diferente de NULL. Após isso, 'pInt' começa a apontar para outro endereço de memória e é impresso o endereço de 'pInt' e seu conteúdo. Também é verificado se 'pInt' é igual a NULL. Após isso, caso ‘ERROR2’ não esteja definido, tenta-se alterar o conteúdo de 'pInt', que gera um erro, pois o caractere '7' não é um endereço de memória válido. A principal diferença entre os dois blocos de código é a inicialização de 'pInt'. Se ‘ERROR2’ não for definido, ‘pInt’ é inicializado com NULL, e caso ‘ERROR2’ for definido, ‘pInt’ não é inicializado, alterando assim as impressões das verificações de estado da variável 'pInt'