Universidade Federal de Ouro Preto - Campus Ouro Preto

Lívia Stéffanny de Sousa - 20.1.4029

BCC322 - Engenharia de Software I

LISTA II – REVISÃO DA LINGUAGEM C

Ouro Preto
2022

Lívia Stéffanny de Sousa - 20.1.4029

LISTA II – REVISÃO DA LINGUAGEM C

.

Professor Orientador: Tiago G. S. Carneiro

Ouro Preto
2022

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

R: “call-back function” é uma função de retorno de chamadas. Você faz uma chamada, ela executa uma instrução, e te dá um retorno.

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

R: É um código para a exibição de uma janela, com as propriedades da biblioteca “windowns.h”. É instanciado uma struct do tipo ‘WNDCLASSEX’ que com ela é setado os parâmetros de definição das propriedades da janela, tais como: tamanho, tipo do cursor, local na tela, style, ícones e etc. Quando o programa está em execução, o código, dentro do seu laço, manipula os eventos de mensagens através de uma ‘call-back function’, CALLBACK WindowProcedure(...), que retorna uma janela com uma mensagem, no caso de o usuário clicar no botão de fechar.

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

R: Ele recebe uma entrada via teclado, se essa entrada for igual a ‘s’, o programa para sua execução. Caso contrário, ele transforma a entrada em letra maiúscula.

4. Como este código funciona?

R: A entrada, caractere, é convertido para inteiro que é o valor correspondente a ele na tabela ASCII. Logo em seguida, é deslocado -32 e exibido o caractere correspondente com o deslocamento.

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

R: O ‘type casting’ é usado para transformar determinada variável em determinado tipo. No exemplo, citado, é necessário usar para fazer a conversão do caractere em inteiro e vice-versa.

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

R: É o número 255.

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

R: Ele da print no ponteiro que foi apontado para uma cadeia de caracteres. Ou, seja, o ponteiro pegou a referência, do endereço de memória, da cadeia de caracteres. Por ser um ponteiro é necessário usar a aritmética de ponteiro para percorrê-lo. Por isso, é incrementado o ponteiro.

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

R: Ele printa os valores, mais o caractere ', ', mais os endereços de memória. Ele printa o ponteiro que recebeu a referência de memória de um array do tipo int. Usa a aritmética de ponteiro para percorrer.

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

R: Ele não funciona, pois não é possível realizar o casting. Ele está tentando fazer um cast de int, que por padrão um ponteiro de int tem 4 bytes, para um ponteiro de char, que tem como padrão 1 byte. Só é possível fazer um cast assim, se fosse do tipo void, que tem por padrão 8 bytes.

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

R: Que com o ponteiro temos o endereçamento de memória para onde ele aponta. Quando se trata de,vetores, deslocamento de posições é incrementado diretamente no ponteiro para acessar suas posições de memória sua operação. E cada tipo de ponteiro tem seu espaço de memória reservado, sendo assim é necessário ficar atendo na realização de cast.

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, baseado na saída impressa na tela, explique o que o trecho de código de linha 27 a 32 faz.

R: Com a diretiva de ERRO1 ativada, ele atribui um novo valor de memória para o ponteiro. Logo sem seguida, ele printa o endereço de memória do ponteiro e o conteúdo. Por fim, ele da falha de segmentação, tenta acessar um endereço inexistente.

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.

R: Pode se observar que mesmo um ponteiro declarado como nulo, null, é reservado um espaço de memória para ele. E declarando um ponteiro, sem inicializar ele como null, ele pode conter lixo ou algum valor já declarado