Marcos Geraldo Braga Emiliano 19.1.4012

Lista 2

1)O call-back é usado como uma função que recebe outra função como parâmetro e executa ela.

2)O programa cria uma janela, seguindo uma serie de critérios que foram passados no programa, e descriptografa alguma chave que foi passada pelo usuário.

3)O código contido no arquivo main1.cpp imprime na tela o caractere maiúsculo de uma letra que foi digitada juntamente com o valor 32, mas como não há tratamento, ele imprime precisamente o caractere 32 posições atras na tabela ASCII, o que pode ser qualquer caractere.

4)Esse código “transforma” o caractere em inteiro e subtrai 32 deste valor, para posteriormente transformar em um caractere novamente.

5)É necessário utilizar-se do casting pois é transformado um caractere em inteiro e posteriormente em caractere e esta é uma forma muito pratica de navegar através da tabela ASCII.

6)“0xFF” é utilizado quando se deseja obter o bit menos significante, mas nesse caso nada é feito, pois ele é apenas passado para um cout.

7)O primeiro looping do main2.cpp imprime a mensagem “Técnicas de Programação II” por meio da aritmética de ponteiros, essa string é passada para um ponteiro de char, onde seu conteúdo é printado a cada iteração, até atingir o \0 no fim da string.

8)O segundo looping do main2.cpp imprime os inteiros de 1 à 9, juntamente com seus os endereços de memoria até atingir o inteiro 9, e após isso imprime na tela a mensagem "----- sizeof(int): " e o valor de sizeof(int).

9)O terceiro looping do main2.cpp deveria aparentemente imprimir os caracteres da tabela ASCII de 0 à 9, porem isso não acontece, e ele não funciona, pois o casting não acontece de forma correta, quando é usada a aritmética de ponteiros neste contexto ocorre um acesso feito de forma equivocada.

10)A aritmética de ponteiros pode ser extremamente útil quando se tem vontade de percorrer um vetor, de forma a sempre ir para a próxima posição obtendo-se o próximo item, ou até mesmo retornar utilizando-se de sucessivas subtrações, o único ponto a ser observado é a necessidade de a memoria ter sido alocada de forma continua.

Tive problemas para compilar os códigos, então vou fazer apenas a análise dos códigos.

11)Quando a diretiva “ERRO1” é ativada é impresso na tela a mensagem “Static”, as variáveis estáticas do tipo inteiro “teste” e “teste3” recebem 1, “variável” recebe 65, das linhas 27 à 32, é primeiro feito um casting de um endereço de memória para um unsigned int, apenas assume valores entre 0 e 256, logo em seguida é impresso, “&pInt: ” o endereço de pInt, logo em seguida “, pInt:” e o conteúdo de pInt, sem seguida após esta impressão á feito uma atribuição onde o conteúdo de pInt se torna 65, em seguida e impresso “&variável: ”, o endereço da variável “variavel”, na mesma linha é impresso “, variável: “ e o casting do valor de variável para um caractere.

12) Quando a diretiva “ERRO2” é ativada o ponteiro unsigned int pInt aponta para nulo, é impresso na tela “not ERRO2”, na linha abaixo é impresso “&pInt: ” em seguida o endereço de pInt, na mesma linha é impresso “, pInt: ”, e é impresso pInt, no caso seria impresso o endereço de memoria que ele aponta, mas colo ele aponta para NULL não há um endereço de memória, na mesmo linha é impresso “, Valid? ”. Na linha 44 é feito uma operação ternaria, onde se pInt for considerado verdadeiro será impresso “TRUE” e do contrário “False”, e na linha debaixo é feita outra comparação ternaria desta vez com o a condicional (pInt =! NULL), e da mesma forma que na linha anterior e impresso “TRUE” em caso verdade e “False” do contrário, na linha 46 pInt recebe um endereço de memória, na linha abaixo é impresso “&pInt: ” seguido do endereço de pInt, “, pInt: ” seguido do espaço de memória presente em pInt, e impresso também ao final “, ” e “ -> ”, na linha abaixo é novamente feita um operação ternaria que imprime “TRUE” caso a condição (pInt != NULL), e falso do contrário, e no fim disto o conteúdo de pInt recebe o caractere ‘7’.

E quando a diretiva esta desativada um else é executado executando a partir da linha 53, é criado um ponteiro de um unsigned int, e impresso “ERRO2”, na linha de baixo é impresso, “&pInt: “seguido do endereço de pInt e “, pInt: “, seguido do endereço de memoria presente em pInt e “, Valid?”, na mesma linha é impresso “TRUE” caso a operação ternaria que tem como condicional (pInt) seja verdadeira e “False” do contrário, na linha abaixo, é feita outra operação ternaria, mas desta vez com a condicional (pInt!=NULL), imprimindo “TRUE” em caso verdadeiro ou “FALSE” do contrário, na linha abaixo pInt recebe para apontar um endereço de memória, é impresso “&pInt: “ logo em seguida é impresso o endereço de memoria de pInt, na mesma linha e impresso “pInt: “ juntamente com o endereço que está em pInt e na mesma linha e impresso “, “ e “ -> “, na ultima linha desta seção, linha 60, é impresso “TRUE” caso a condicional (pInt!= NULL) seja verdadeira e “FALSE” do contrário.