Lista 2 - Engenharia de software

Matheus Peixoto Ribeiro Vieira - 22.1.4104

main.cpp

1 - É uma função que é executada sempre que algum evento acontecer ou um código chegar a um determinado estado, sendo que a mesma é passada como parâmetro de uma função. Geralmente ela é assíncrona, ou seja, fica em espera até que sua execução seja liberada.

2 - O código cria uma janela usando a API do windows.

A função WinMain instancia um objeto da classe WNDCLASSEX, que define características da janela, como o nome, a função WindowProcedure para ser chamada pelo windows, definir o reconhecimento de cliques duplos, reconhecimento dos ícones do mouse e o uso das cores padrão do windows. Depois instância um CreateWindowEx com as características visíveis da janela, como o seu tamanho e exibe a mesma com o ShowWindow. Depois inicia um loop que fica recebendo mensagens até que o valor da mesma seja zero. As mensagens recebidas são traduzidas pelo TranslateMessage e enviadas para o windows enviar ao WindowProcedure pelo DispatchMessage para que sejam processadas.

A função WindowProcedure é uma função de callback que sempre é executada quando há uma mensagem para a janela. No caso do código, ela é chamada quando a janela é fechada, lidando com a mensagem ‘WM_DESTROY’, mandando uma WM_QUIT para a fila de mensagens do programa.

main1.cpp

3 - O código contido no arquivo main1.cpp recebe como entrada um valor de um caractere escrito pelo usuário, imprime na tela a diferença inteira entre os valores de ‘A’ e ‘a’ da tabela ASCII e, depois exibe na tela, o caractere correspondente ao que foi escrito menos 32 na tabela ASCII.

4 - O código contido no arquivo main1.cpp imprime na tela uma mensagem de “Hello World”, depois instância uma variável `c` do tipo char e um `incr` do tipo int.

Em sequência, inicia um loop que tem, como condição de parada, o valor da variável `c` ser a letra ‘s’.

Dentro do loop, é lido, do terminal, um caractere qualquer escrito pelo usuário, atribuindo o valor à variável `c`. Em seguida, calcula o valor entre os caracteres ‘A’ e ‘a’, converte para inteiro, sendo o resultado -32, salva na variável incr e imprime-a.

Depois converte o caractere que foi escrito para um valor, na tabela ASCII, que é o mesmo decrescido de 32. Dessa forma, uma letra minúscula vira uma maiúscula e uma maiúscula vira algum símbolo.

5 - O primeiro type cast foi necessário para converter o valor a ser calculado de um caractere para inteiro, todavia ele não seria necessário, uma vez que, pelo fato de incr ser um inteiro, ela já utilizaria o valor numérico. Mas o segundo type cast é necessário porque a diferença de c - 32 é retornada como um inteiro, porém deseja-se imprimir o caracter correspondente a esse número, logo, deve-se realizar o cast

main2.cpp

6 - O valor 0xFF representa, simplesmente, o valor “255” em base hexadecimal.

7 - A variável pChar, antes do loop, recebe o endereço do primeiro item do vetor de caracteres disciplina. Assim, dentro do loop, pChar está tendo o conteúdo que ele referencia sendo impresso na tela e indo para o próximo endereço de memória. Sendo que isso se repete até que o conteúdo do endereço de memória que pChar se refere é o caractere indicador de finalização de uma string.

8 - O segundo loop funciona a partir de um do while, dessa forma, ele sempre realiza a primeira iteração. Assim, ele exibe na tela o conteúdo apontado por pInt juntamente com o seu valor, que é um endereço de memória. Depois incrementa mais um ao pInt, fazendo com que ele aponte para o próximo endereço. Assim, este loop funciona até que o valor referenciado por pInt seja igual a 9.

9 - O terceiro loop tinha a intenção de percorrer todos os 10 valores do vetor de algarismos a partir de um for loop, enquanto mostrava o conteúdo de cada item e

seu endereço de memória. Todavia, o código não funciona corretamente pois está tentando ser realizado um cast de um ponteiro de inteiro para char, sendo que o tamanho de um char é de 1 byte e um inteiro são 4 bytes. Dessa forma, os dados não são lidos corretamente.

10 - Podemos concluir que a aritmética de ponteiros funciona corretamente quando é realizada a partir dos tipos de dados corretos, porém é necessário tomar cuidado com relação ao uso incorreto dos tipos de dados, pois, nem sempre, o compilador avisará sobre possíveis erros de aritmética de ponteiros.

main3.cpp

11 - O trecho de código da linha 27 a 32 cria um ponteiro para um endereço de memória específico, exibe este endereço e o valor contido no mesmo. Depois, atualiza o valor que está no endereço de memória e exibe, novamente, o endereço e o seu valor, porém, como foi atribuído um inteiro conhecido, o valor a ser exibido é convertido para char.

12 - O trecho de código entre a linha 37 e 57 inicia um ponteiro de inteiro sem sinal com um valor nulo. Depois disso, exibe na tela “not ERRO2” e, em sequência, o endereço do ponteiro, o valor que está no endereço de memória e, se pInt for igual a zero, exibe FALSE, caso contrário, TRUE. Como pInt é NULL, então é exibido o FALSE. Em sequência, como pInt é nulo, exibe “FALSE”.

Em sequência, pInt é apontado para um endereço de memória específico e ocorre a exibição do endereço do ponteiro, o valor que ele referencia e TRUE ou FALSE caso o ponteiro seja nulo.

Por fim, o valor apontado por pInt é atribuído como o caractere 7, mas como o valor é um char, será convertido para o seu valor inteiro da tabela ascii

Por outro lado, o código com ERROR2 definido realiza operações semelhantes, mas não estamos com o pInt definido como NULL, dessa forma, o código tenta acessar um valor que não foi definido, podendo fazer com que oocorra um *crash* no código.

Como pInt simplesmente aponta para um endereço de memória, o seu valor é desconhecido e, por isso, a exibição do seu valor pode ser qualquer coisa. Essa indeterminação também ocorre para o print de valid e se pInt é diferente de nulo.

Depois, pInt irá apontar para um endereço de memória específico e terá o seu endereço exibido na tela, junto com o valor que está naquela posição de memória.

Como agora ele não é mais nulo, será exibido TRUE