UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

CIÊNCIA DA COMPUTAÇÃO

ENGENHARIA DE SOFTWARE I

LISTA 2

Marcus Vinícius Souza Fernandes

19.1.4046

Ouro Preto

2021

1

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

O código do arquivo main.cpp realiza alterações nas janelas do windows utilizando a biblioteca windows.

3

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

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

O *type casting* 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 *type casting* será necessário novamente para converter o valor numérico resultante em um caractere.

6

O valor 0xFF no arquivo main2.cpp representa o valor 255 no formato hexadecimal.

7

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 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 terceiro loop do código presente em main2.cpp deveria imprimir o conteúdo de um vetor de caracteres obtido através de um *type casting* feito a partir de um vetor de inteiros, porém isso não acontece pois há uma incompatibilidade entre o tipo do ponteiro (char) e o tipo presente dentro do vetor (int). Vale lembrar que o tamanho de um tipo char ocupa menos espaço que um inteiro em memória, e deste modo um ponteiro de char é insuficiente para representar um tipo inteiro.

10

Quanto à aritmética de ponteiros é possível concluir que ao realizar operações sobre os ponteiros é necessário que o tamanho dos tipos envolvidos sejam semelhantes. Em especial, no código estudado, foi possível observar os erros obtidos quando é realizado um *type casting* entre tipos não equivalentes (inteiro e char).

11

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

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'.