ANO
2024

UNINTER

PROGRAMAÇÃO ESTRUTURADA

LISTA DE EXERCÍCIOS – AULA PRÁTICA 1

Prof. Winston Sen Lun Fung, Me.

Exercício 01

Uma livraria deseja desenvolver um sistema para cadastrar livros. Cada livro possui um título, um autor e o ano de publicação. No entanto, o sistema deve garantir que os usuários não insiram mais caracteres do que os permitidos nos campos de título e autor (100 e 50 caracteres, respectivamente). Implemente o sistema que permita o cadastro de 3 livros e garanta que as entradas do usuário não excedam os limites permitidos. Em seguida, exiba as informações dos livros cadastrados.

Solução sugerida:

```c
#include <stdio.h>  // Biblioteca padrão para entrada e saída de dados

int main() {
    // Definição de um array de strings (produtos), um array de inteiros (quantidade) e um array de floats (preços)
    char produtos[3][50];   // Armazena até 10 produtos, cada um com até 49 caracteres (50 incluindo o caractere nulo)
    int quantidade[3];      // Armazena a quantidade de cada produto
    float precos[3];        // Armazena os preços de cada produto
    float valorTotal = 0.0;  // Variável para armazenar o valor total do estoque
    int i, escolha, qtdaVenda;  // Variáveis auxiliares para contagem e para a escolha e quantidade de vendas
    char c;  // Variável usada para capturar o enter após o uso de scanf_s()

    // Solicitar a entrada de dados dos produtos
    for (i = 0; i < 3; i++) {  // Loop para cadastrar 3 produtos
        printf("Digite o nome do produto %d: ", i + 1);  // Solicita o nome do produto
        gets_s(produtos[i], 49);  // Função para capturar o nome do produto, limitada a 49 caracteres

        printf("Digite a quantidade do produto %d: ", i + 1);  // Solicita a quantidade do produto
        scanf_s("%d", &quantidade[i]);  // Lê a quantidade e armazena no array quantidade
        c = getchar();  // Captura o enter deixado pelo scanf_s para evitar que interfira na próxima entrada

        printf("Digite o preco do produto %d: ", i + 1);  // Solicita o preço do produto
        scanf_s("%f", &precos[i]);  // Lê o preço e armazena no array precos
        c = getchar();  // Captura o enter deixado pelo scanf_s

        printf("\n");  // Adiciona uma linha em branco para separar a saída
    }

    // Mostrar os produtos em estoque
    printf("\nEstoque de produtos:\n");

    for (i = 0; i < 3; i++) {  // Loop para exibir os 3 produtos cadastrados
        printf("Produto: %s \t Qtda: %d \t\t Preco: %.2f \n", produtos[i], quantidade[i], precos[i]);  // Exibe os detalhes do produto
        valorTotal += quantidade[i] * precos[i];  // Calcula o valor total do estoque
```

```
34.     }
35.
36.     printf("\nO Valor total do estoque eh: %.2f\n", valorTotal);  // Exibe o valor total do estoque
37.
38.     // Alteração no estoque após a venda
39.     printf("\nDigite o codigo do produto vendido [1 - 10]: ");  // Solicita o código do produto vendido (indexado de 1 a 10)
40.     scanf_s("%d", &escolha);  // Lê o código do produto escolhido
41.     c = getchar();  // Captura o enter
42.
43.     printf("\nQual a quantidade vendida: ");  // Solicita a quantidade vendida
44.     scanf_s("%d", &qtdaVenda);  // Lê a quantidade de venda
45.     c = getchar();  // Captura o enter
46.
47.     quantidade[escolha - 1] = quantidade[escolha - 1] - qtdaVenda;  // Subtrai a quantidade vendida do estoque
48.
49.     valorTotal = 0.0;  // Reseta o valor total para recalculá-lo
50.
51.     // Atualiza e mostra o novo estoque após a venda
52.     for (i = 0; i < 3; i++) {  // Loop para exibir os 3 produtos atualizados
53.         printf("Produto: %s \t Qtda: %d \t\t Preco: %.2f \n", produtos[i], quantidade[i], precos[i]);
54.         valorTotal += quantidade[i] * precos[i];  // Recalcula o valor total do estoque
55.     }
56.
57.     printf("\nO Valor total do estoque eh: %.2f\n", valorTotal);  // Exibe o novo valor total do estoque
58.
59.     return 0;  // Encerra o programa
60. }
61.
```

Explicação Detalhada:

1. **Uso do `gets_s`**: A função `gets_s` é usada para ler strings (nome do produto) com segurança, garantindo que não ocorra overflow. O limite de caracteres é controlado pelo segundo parâmetro (49).
2. **Tratamento do enter (`getchar`)**: Após o uso de `scanf_s`, a função `getchar()` captura o caractere de nova linha deixado no buffer, evitando que interfira em leituras futuras de strings.
3. **Cálculo do valor total**: O valor total do estoque é calculado multiplicando a quantidade de cada produto pelo seu preço e somando todos os resultados.
4. **Manipulação de estoque**: Quando um produto é vendido, a quantidade no vetor (array) é atualizada, e o valor total do estoque é recalculado.

Exercício 02

Uma empresa deseja criar um sistema para cadastrar funcionários. Cada funcionário tem um nome, um cargo e um salário. O nome pode ter no máximo 50 caracteres e o cargo pode ter até 30 caracteres. Após cadastrar 5 funcionários, o sistema deve verificar se há algum funcionário com o mesmo cargo e, se houver, exibir os detalhes desses funcionários.

Solução sugerida:

```c
#include <stdio.h>
#include <string.h>

int main() {
    // Declaração da matriz "cinema" que representa a sala de cinema com 5 fileiras e 5 colunas
    char cinema[5][5];
    int i, j, fileira, coluna; // Variáveis para controle de loops e coordenadas dos assentos
    char comando[5], c;        // Variáveis para capturar o comando e tratar o buffer de entrada

    // Inicializa a sala de cinema com todos os assentos desocupados ('O' significa assento livre)
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            cinema[i][j] = 'O'; // Todos os assentos começam como disponíveis
        }
    }

    // Loop principal do programa para gerenciamento das reservas
    while (1) {
        // Exibe o estado atual do cinema (assentos disponíveis e reservados)
        for (i = 0; i < 5; i++) {
            for (j = 0; j < 5; j++) {
                printf("%c ", cinema[i][j]); // Exibe 'O' para assento livre e 'X' para assento ocupado
            }
            printf("\n"); // Move para a próxima fileira após imprimir a atual
        }

        // Solicita ao usuário um comando para encerrar ou continuar o programa
        printf("\n\nDigite SAIR para encerrar o programa: ");
        fgets(comando, 4, stdin); // Lê o comando digitado pelo usuário

        comando[4] = '\0'; // Garante que a string de comando termine com '\0' (caractere nulo)

        // Verifica se o comando é "SAIR" para encerrar o programa
        if (strcmp(comando, "SAIR") == 0) {
            break; // Sai do loop se o usuário digitar "SAIR"
```

```
36.         }
37.
38.         // Solicita ao usuário as coordenadas do assento desejado
39.         printf("\n\nDigite a fileira [1 - 5] e a coluna [1 - 5] do assento desejado: ");
40.         scanf_s("%d %d", &fileira, &coluna); // Lê a fileira e a coluna que o usuário quer reservar
41.         c = getchar(); // Captura o caractere de nova linha '\n' após o scanf_s para limpar o buffer
42.
43.         // Verifica se as coordenadas inseridas estão dentro do intervalo permitido
44.         if (fileira < 1 || fileira > 5 || coluna < 1 || coluna > 5) {
45.             printf("\n\nDigite as coordenadas corretas! Tente novamente.\n");
46.             continue; // Retorna ao início do loop caso as coordenadas sejam inválidas
47.         }
48.
49.         // Verifica se o assento escolhido está disponível ('O') ou já ocupado ('X')
50.         if (cinema[fileira - 1][coluna - 1] == 'O') {
51.             cinema[fileira - 1][coluna - 1] = 'X'; // Marca o assento como ocupado ('X')
52.             printf("\nAssento %d,%d reservado com sucesso.\n", fileira, coluna); // Confirmação da reserva
53.         }
54.         else {
55.             printf("\nAssento já está reservado.\nTente Novamente.\n"); // Informa que o assento já está ocupado
56.         }
57.     }
58.
59.     // Mensagem final ao encerrar o programa
60.     printf("\n\nObrigado por utilizar o nosso sistema! Ateh logo.\n\n");
61.
62.     return 0; // Finaliza o programa
63. }
64.
```

Explicação Detalhada:

1. **Declaração de Variáveis**:
   - `cinema[5][5]`: Uma matriz 5x5 que simula a sala de cinema, onde cada elemento representa um assento.
   - `fileira` e `coluna`: Armazenam as coordenadas do assento que o usuário deseja reservar.
   - `comando[5]`: String para capturar o comando digitado pelo usuário (usada para verificar se o comando "*SAIR*" foi inserido).
   - `c`: Variável auxiliar para capturar o caractere de nova linha após o uso de `scanf_s`, evitando problemas no buffer de entrada.
2. **Inicialização da Matriz**:
   - A matriz cinema é inicializada com o caractere '`O`', indicando que todos os assentos estão livres no início do programa.
3. **Loop Principal**:
   - O programa exibe o estado atual da matriz (cinema), mostrando quais assentos estão livres ('`O`') e quais estão ocupados ('`X`').
   - O usuário pode digitar "*SAIR*" a qualquer momento para encerrar o programa.

- Se o usuário não digitar "*SAIR*", o programa solicitará a fileira e a coluna do assento que deseja reservar.

4. **Validação das Coordenadas**:
   - O programa verifica se as coordenadas digitadas pelo usuário estão dentro dos limites permitidos (fileiras e colunas de 1 a 5).
   - Caso as coordenadas sejam inválidas, o programa pede ao usuário que tente novamente.
5. **Reserva de Assentos**:
   - Se o assento escolhido estiver livre ('O'), o programa o marca como ocupado ('X') e exibe uma mensagem de confirmação.
   - Se o assento já estiver ocupado ('X'), o programa avisa ao usuário e solicita que ele tente novamente.
6. **Encerramento**:
   - Quando o usuário digita "*SAIR*", o programa sai do loop principal e exibe uma mensagem de despedida.

Exercício 03

Uma escola deseja criar um sistema para cadastrar alunos e verificar se são maiores de idade. Cada aluno tem um nome e uma idade. O nome pode ter no máximo 50 caracteres. Após cadastrar 4 alunos, o sistema deve exibir os nomes e idades dos alunos maiores de idade (18 anos ou mais).

Solução sugerida:

```c
#include <stdio.h>
#include <string.h>

// Função que substitui todas as ocorrências de uma palavra errada por uma palavra correta em um texto
void substituirPalavra(char texto[], char palavraErrada[], char palavraCorreta[], int* contagem) {
    char resultado[1000]; // String temporária para armazenar o texto corrigido

    // Ponteiros para manipulação do texto
    char* posicao, * inicioTexto = texto;

    // Calcula o tamanho das palavras errada e correta
    int tamErrada = strlen(palavraErrada);
    int tamCorreta = strlen(palavraCorreta);

    // Inicializa a string resultado como vazia
    resultado[0] = '\0';

    // Laço que busca e substitui todas as ocorrências da palavra errada
    while ((posicao = strstr(inicioTexto, palavraErrada)) != NULL) {
        // Copia para o resultado a parte do texto antes da palavra errada
        strncat_s(resultado, sizeof(resultado), inicioTexto, posicao - inicioTexto);

        // Adiciona a palavra correta no resultado
        strcat_s(resultado, sizeof(resultado), palavraCorreta);

        // Avança o ponteiro de início do texto após a palavra errada encontrada
        inicioTexto = posicao + tamErrada;

        // Incrementa o contador de substituições
        (*contagem)++;
    }

    // Copia o restante do texto que não contém a palavra errada
    strcat_s(resultado, sizeof(resultado), inicioTexto);

```

```
36.     // Copia o conteúdo de 'resultado' de volta para 'texto', substituindo-o
37.     strcpy_s(texto, sizeof(resultado), resultado);
38. }
39.
40. int main() {
41.     char texto[1000], palavraErrada[100], palavraCorreta[100]; // Strings para armazenar o texto, a palavra errada e a palavra correta
42.     int contagem = 0; // Variável para contar o número de substituições feitas
43.
44.     // Solicita ao usuário que insira um parágrafo de texto
45.     printf("Digite um parágrafo de texto: ");
46.     fgets(texto, 999, stdin); // Lê até 999 caracteres de texto (deixando espaço para o caractere nulo '\0')
47.
48.     // Solicita ao usuário a palavra incorreta que ele deseja substituir
49.     printf("\n\nDigite a palavra incorreta: ");
50.     scanf_s("%s", palavraErrada, 99); // Lê a palavra incorreta, limitando a 99 caracteres
51.
52.     // Solicita ao usuário a palavra correta que substituirá a palavra errada
53.     printf("\n\nDigite a palavra correta: ");
54.     scanf_s("%s", palavraCorreta, 99); // Lê a palavra correta, limitando a 99 caracteres
55.
56.     // Chama a função substituirPalavra para realizar as substituições no texto
57.     substituirPalavra(texto, palavraErrada, palavraCorreta, &contagem);
58.
59.     // Exibe o texto corrigido após a substituição
60.     printf("\n\nTexto corrigido: \n%s", texto);
61.
62.     // Exibe o número de palavras substituídas
63.     printf("\nnumero de palavras substituidas: %d \n", contagem);
64.
65.     return 0; // Finaliza o programa
66. }
67.
```

Explicação Detalhada:

1. **Bibliotecas Incluídas**:
   - `#include <stdio.h>`: Biblioteca padrão de entrada e saída, que fornece funções como `printf`, `scanf_s`, `fgets`.
   - `#include <string.h>`: Biblioteca para manipulação de strings, com funções como `strstr`, `strlen`, `strcat_s`, `strncat_s`, e `strcpy_s`.
2. **Função substituirPalavra**:
   - **Parâmetros**:
     - `char texto[]`: O texto no qual a substituição ocorrerá.
     - `char palavraErrada[]`: A palavra que será substituída.

- `char palavraCorreta[]`: A palavra que substituirá a palavra errada.
- `int* contagem`: Um ponteiro para uma variável inteira que conta quantas vezes a substituição foi feita.

- **Variáveis Locais**:
  - `resultado[1000]`: String temporária onde o texto corrigido será montado.
  - `char* posicao, * inicioTexto = texto`: Ponteiros para localizar a palavra errada dentro do texto.
  - `tamErrada e tamCorreta`: Armazenam os tamanhos das palavras errada e correta, respectivamente, utilizando a função `strlen`.
- **Lógica**:
  - O laço `while` usa `strstr` para localizar a próxima ocorrência da palavra errada no texto.
  - Cada vez que a palavra errada é encontrada, a parte anterior do texto é copiada para resultado, seguida pela palavra correta.
  - O ponteiro `inicioTexto` é atualizado para pular a palavra errada e continuar a busca até o final do texto.
  - Ao final, `strcpy_s` é usado para copiar o texto corrigido de resultado de volta para texto.

3. **Função main**:
   - **Variáveis**:
     - `texto[1000]`: Armazena o parágrafo de texto inserido pelo usuário (máximo de 999 caracteres + caractere nulo).
     - `palavraErrada[100]`: Armazena a palavra que será substituída.
     - `palavraCorreta[100]`: Armazena a palavra que será usada como substituta.
     - `contagem`: Variável que conta quantas substituições foram realizadas.
   - **Passos**:
     1. O programa solicita ao usuário que insira um parágrafo de texto utilizando `fgets`, que lê até 999 caracteres e garante que o texto não exceda o limite.
     2. O usuário é solicitado a inserir a palavra errada e a palavra correta.
     3. A função `substituirPalavra` é chamada para processar o texto.
     4. O texto corrigido é exibido, e o número de substituições realizadas é mostrado na tela.

Pontos Importantes:

- **Segurança com Strings**: O código utiliza as funções seguras `scanf_s`, `strncat_s`, `strcat_s`, e `strcpy_s` para evitar o overflow de buffer, protegendo o programa contra possíveis vulnerabilidades.
- **Limpeza do Buffer**: `fgets` é usada para capturar a entrada do texto, o que garante que a entrada não ultrapasse o limite do array.
- **Ponteiros e Manipulação de Strings**: O uso de `strstr` permite localizar a palavra errada no texto, e o ponteiro `inicioTexto` avança após cada substituição, garantindo que o texto seja percorrido corretamente.
- **Modularidade**: A lógica de substituição está separada na função `substituirPalavra`, tornando o código mais modular e fácil de entender.