INF 1037 - Programação em C - Turmas 33A - 29/09/2025
Profa. Paula Rodrigues
Lista de Exercícios 06: Caractere, Cadeia de Caracteres e Vetores de Cadeia de Caracteres

PARTE 1:
Observações:

- **As questões 01 a 09 foram retiradas do livro "Introdução a Estruturas de Dados - Com Técnicas de Programação em C (Portuguese Edition)" - Waldemar Celes.**
- **Para todas as questões abaixo, faça apenas uma função `main`. As chamadas às funções auxiliares para testá-las devem ser TODAS chamadas desta mesma `main`.**

Questão 01) Implemente uma função que receba como parâmetros uma cadeia de caracteres (string) e um caractere c, e retorne o número de ocorrências do caractere dentro da cadeia. Por exemplo, se forem passados para a função a cadeia "Rio de Janeiro" e o caractere 'i', a função deve retornar o valor 2. A função deve obedecer ao protótipo a seguir:

```
int conta_ocorrencias (char* s, char c);
```

Questão 02) Implemente uma função que receba uma string como parâmetro e retorno como resultado o número de vogais nesta string. Essa função deve obedecer ao protótipo:

```
int conta_vogais (char* s);
```

Questão 03) Implemente uma função que receba uma string como parâmetro e altere nesta string as ocorrências de caracteres maiusculos para minúsculos. Essa função deve obedecer ao protótipo:

```
void minusculo (char* s);
```

Questão 04) Implemente uma função que receba uma string como parâmetro e substitua todas as letras por suas sucessoras no alfabeto. Por exemplo, a string "Casa" seria alterada para "Dbtb". A letra z deve ser substituída pela letra a (e Z por A). Caracteres que não forem letras devem permanecer inalterados. Essa função deve obedecer ao protótipo:

```
void shift_string (char * str);
```

Questão 05) Implemente uma função que receba uma string como parâmetro e substitua as ocorrências de uma letra pelo seu oposto no alfabeto, isto é, a ↔ z, b ↔ y, c ↔ x etc. Caracteres que não forem letras devem permanecer inalterados. Essa função deve obedecer ao protótipo:

```
void string_oposta (char * str);
```

Questão 06) Implemente uma função que receba uma string como parâmetro e desloque os seus caracteres uma posição para a direita. Por exemplo, a string "casa" seria alterada para "acas". Repare que o último caractere vai para o início da string. Essa função deve obedecer ao protótipo:

```
void roda_string (char * str);
```

Questão 07) Implemente uma função que receba como parâmetros uma cadeia de caracteres (string) e um número inteiro n, e retorne uma nova cadeia, alocada dinamicamente dentro da função, que represente a cadeia original sem os últimos n caracteres. Por exemplo, se forem passados para a função a cadeia " Rio de Janeiro " e o número 4, a função deve retornar a cadeia " Rio de Jan ". A função deve obedecer ao protótipo a seguir.

```
char * retira_sufixo (char * s, int n);
```

Assuma que a cadeia original terá sempre mais do que *n* caracteres.

Questão 08) Escreva uma função em C que receba como parâmetro uma string e retorne uma nova string com somente as letras do alfabeto (serão removidos caracteres especiais, espaços, dígitos etc.). Por exemplo, se for passada como parâmetro a cadeia de caracteres "# Mat .: 39838-0 DC ", a função deve retornar a cadeia " MatDC ". A string passada como parâmetro não pode ser alterada. Essa função deve ter o seguinte protótipo:

```
char * converte (char * s);
```

Questão 09) Escreva uma função que receba como parâmetros duas strings e um caractere separador. A função deve criar a string que representa a concatenação das duas strings de entrada, usando o caractere como separador. Por exemplo, se forem passadas as strings “ex” e “aluno”, e o caractere hífen ‘-’, deve-se ter como valor de retorno a string “ex-aluno”. O protótipo da função deve ser:

```
char * concatena (char * s1 , char * s2 , char sep);
```

PARTE 2:

Questão 10) Escreva um programa que use a função `strcmp` para comparar duas strings digitadas pelo usuário. O programa deverá indicar se a primeira string é menor, igual ou maior que a segunda.

Questão 11) Escreva um programa que leia a idade e o primeiro nome de, no máximo, 10 pessoas. Se o usuário fornecer uma idade negativa, seu programa deve interromper a leitura. No final da execução, devem ser exibidos o nome e a idade da pessoa mais jovem e da pessoa mais velha.

Questão 12) Escreva um programa que recebe do usuário uma string *s*, um caractere *c* e uma posição (índice) *i* e devolve o índice da primeira posição da string onde foi encontrado o caractere *c*. A procura deve começar a partir da posição *i* inclusive.

Questão 13) Escreva um programa em que troque todas as ocorrências de uma letra L1 pela letra L2 em uma string. A string e as letras L1 e L2 devem ser fornecidas pelo usuário.

Questão 14) Escreva uma função que recebe uma matriz *m* x *n* de caracteres contendo strings de tamanhos variáveis em cada linha e converta-a em um vetor de ponteiros para caracteres alocando dinamicamente o espaço em memória para cada string a ser inserida no vetor. Não esqueça de testar a alocação e liberar a memória após seu uso. Crie um novo programa (main) que teste/use a função criada.

Questão 15) Escreva uma função que recebe um vetor de inteiros *v* e a quantidade de elementos *n* nesse vetor e cria um vetor de ponteiros para *char*, contendo strings com a seguinte regra de formação. Se *v[i]* for par, a string deverá conter *k* vezes a letra ‘A’. Por outro lado, se *k* for ímpar, a string deverá conter *k* vezes a letra ‘a’. Como exemplo, para um vetor de entrada *v* = { 2, 4, 3, 1 }, é esperado que seja criado dinamicamente um vetor de char* contendo as strings “AA”, “AAAA”, “aaa” e “a”.

```
char ** converte (int * v , int n);
```