Universidade Federal do Espírito Santo

Disciplina de Programação II - 2019/2

Professores: Francisco de Assis S. Santos, Dr./Fernanda Matos.

LISTA DE EXERCÍCIOS IV - 31/10/2019 (Entregar em 15/11/2019)

QUESTÕES SOBRE STRINGS:

1. Fazer um programa em C que lê um string (palavra) do teclado e imprime o string de maneira normal e de trás para diante. Suponhamos, a palavra lida seja "Programacao" o resultado deverá ser: "Programacao" e "oacamargorP".

DICA: Pode-se utilizar um o laço de repetição "for" e controlar dois índices simultaneamente, por exemplo: for (i=0, j=strlen(palavra)-1; i<j; i++, j--) { // comandos da troca de caracteres ... }. Ou seja, o índice i pode ser utilizado para percorrer a partir do início da palavra para frente e o índice j percorrer do final da palavra para o início, permitindo realizar as trocas de caracteres, e quando os índices se encontrarem significa que toda a palavra foi percorrida, em sentidos opostos, pelos dois índices, i e j. (OBS: strlen() é uma função da biblioteca string.h)

2. Implemente um rotina que faça a mesma coisa que a função "strcpy".

3. Fazer um uma rotina em C que lê um string qualquer de no máximo 100 caracteres e imprimir. Elaborar um menu de opções que invoque funções para cada uma das situações abaixo:

(a) Quantos caracteres tem o string;

(b) Quantos caracteres são de pontuação (não é acentuação);

(c) Quantos caracteres são números;

(d) Quantos caracteres são minúsculas;

(e) Quantos caracteres são maiúsculos;

(f) Quantos espaços em branco.

DICAS: Utilizar as funções da biblioteca <ctype.h>, conforme o Quadro a seguir:

Funções booleanas	Descrição
isxdigit(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é compatível com um número hexadecimal.
isalpha(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é alfabético. Isso inclui todas as letras do alfabeto, tanto maiúsculas quanto minúsculas.
isdigit(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é número.
isalnum(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é alfanumérico. Isso inclui todos os números e as letras do alfabeto.
ispunct(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é uma pontuação. Isso inclui qualquer tipo de pontuação como . , ? ! ^ ' { } ~ : ;.
isspace(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é um espaço em branco.
islower(argumento)	Retorna um número inteiro diferente de zero se o caractere passado como parâmetro é uma letra minúscula.
isupper(argumento)	Retorna um número inteiro diferente de zero caso o caractere passado como parâmetro é uma letra maiúscula.

Por exemplo, uma função para calcular a quantidade de caracteres alfanuméricos pode assumir:

```
int contaAlfanumericos( char palavra[  ] ){ /*Passa um vetor de char/String e deve retornar um número inteiro*/

  int i, cont=0;

  for (i=0;i<strlen(palavra);i++)

   if ( isalnum (palavra[i]) )  /* Utiliza a função que verifica se o caractere é
                                   alfanumérico */

      cont+=1; /*Conta a quantidade de caracteres alfanuméricos */

    return cont; /*Depois que sai do laço retorna a quantidade de caracteres alfanuméricos*/

}
```

4. Escreva uma rotina em C, uma void, que receba um string (por exemplo palavra [10]), um caractere (char caractere) e o índice de uma posição do string (int posicao) como parâmetros e insira o carácter na posição informada da palavra "empurrando" todos os demais caracteres para frente. Suponhamos, se a palavra lida for "Programacao", o carácter 'X' e o índice 4, deverá ser obtido como resultado: "ProXgramacao". Invocar a void implementada na void main() e realizar testes.

DICA: Utilizar uma variável do tipo string (char aux[10]) como auxiliar de maneira a facilitar a atribuição do caractere na posição indicada.

5. Construa uma rotina em C capaz de ler duas variáveis do tipo string de até no máximo 90 caracteres (char Frase_A [90] e char Frase_B [90]). A partir disso, estabelecer uma lógica de maneira que: Sendo a Frase_A maior que a Frase_B deve-se copiar a Frase_A para a Frase_B, caso contrário, deve-se concatenar a Frase_B no final da Frase_A.

DICA: Utilizar as funções da biblioteca <string.h>, conforme suas utilidades descritas no quadro abaixo.

Funções de manipulação de strings (A,B)	Descrição
strcmp(A,B)	Realiza a comparação entre duas strings, A e B. Se A for maior que B a função retorna um valor positivo (> 0), se A for menor que B retorna um valor negativo (< 0) e se A e B forem iguais retorna 0.
strcpy(A,B)	Copia a variável string B para a variável A, substituindo o conteúdo atual da variável.
strlen(A)	Retorna a quantidade de caracteres existentes em A, incluindo espaços.
strcat(A,B)	Concatena no final da string A a string B.
strncat(A,B,tamanho)	Concatena ao final da variável A a quantidade de caracteres da variável B especificada em tamanho.
strncmp(A,B,tamanho)	Realiza a comparação entre duas strings, A e B. Se A for maior que B a função retorna um valor positivo > 0, se A for menor que B retorna um valor negativo e se A e B forem iguais retorna 0. Assim como na função strcmp, porém, nessa função há a necessidade de informar o tamanho a ser comparado (ou seja, a quantidade de caracteres a ser considerada em ambas as strings, A e B).
strncpy(A,B,tamanho)	Copia a variável string B para a variável A, substituindo o conteúdo atual da variável. Assim como na função strcpy, porém, deve ser informado o tamanho a ser considerado.

QUESTÕES SOBRE VETORES ESTÁTICOS:

6. Escreva um programa que apresente o seguinte menu e implemente funções para cada opção:
1 - Ler Vetor de inteiros de 10 posições;
2 - Imprimir Vetor;
3 - Exibir apenas os números pares do vetor;

4 - Exibir apenas os números ímpares do vetor;
5 - Exibir quantos números pares existem em posições ímpares do vetor;
6 - Exibir quantos números ímpares existem em posições pares do vetor;
7 – Imprimir o vetor na ordem inversa;
8 – Sair;

7. Desenvolva em C uma estrutura matricial (estática) que permita obter a intersecção entre o valor de vendas (float) de cinco tipos de produtos de um estabelecimento comercial com os meses do ano.

8. Dado o pseudocódigo abaixo, desenhe o fluxograma e desenvolva o código correspondente na linguagem C. Utilize o comando #define para a constante QUANTIDADE_DE_PESSOAS e também um vetor do tipo struct Pessoa.

```
REGISTRO Pessoa
  nome, sexo: TEXTO
  peso, altura, cpf: NUMÉRICO
FIM-REGISTRO
QUANTIDADE_DE_PESSOAS = 3 // constante

pessoas: vetor [0..QUANTIDADE_DE_PESSOAS-1] de REGISTRO Pessoa
 //O vetor pessoas é do tipo de dados de Registro Pessoa, com tamanho
//QUANTIDADE_DE_PESSOAS
cpf_localizador: NUMÉRICO


PARA i=0 ATÉ QUANTIDADE_DE_PESSOAS-1 PASSO 1 FAÇA
  LER pessoas[i].nome
  LER pessoas[i].altura
  LER pessoas[i].peso
  LER pessoas[i].cpf
  LER pessoas[i].sexo
FIM-PARA

LEIA cpf_localizador

PARA i=0 ATÉ QUANTIDADE_DE_PESSOAS-1 PASSO 1 FAÇA
  SE pessoas[i].cpf == cpf_localizador ENTÃO
    ESCREVER pessoas[i].nome
    ESCREVER pessoas[i].sexo
    // Cálculo do IMC = peso / (altura * altura)
    ESCREVER pessoas[i].peso / (pessoas[i].altura * pessoas[i].altura)
 FIM-SE
FIM-PARA
```

9. Para o código em C obtido na resolução da questão anterior, adicionar uma rotina (em um novo arquivo.c) que pergunte ao usuário se deseja adicionar novas pessoas, em caso da resposta seja positiva -SIM-, deve-se solicitar o número adicional de pessoas a serem inseridas. Dessa maneira, o código em C deve ser alterado para alocação dinâmica de memória referente ao vetor pessoas (utilizar o comando malloc) e, posteriormente realocar nova memória de maneira a suportar as novas informações de pessoas a serem adicionadas (utilizar o comando realloc). Sempre ao final do uso do vetor deve-se liberar a memória aplicando o comando free.

10. O pseudocódigo abaixo representa a solução para encontrar o ponto mais próximo do primeiro ponto lido de coordenadas cartesianas. São consideradas cinco leituras de pontos. Esboce o fluxograma correspondente e obtenha a solução na linguagem C.

```
REGISTRO Ponto
  x, y: NUMÉRICO
FIM-REGISTRO

QUANTIDADE_DE_PONTOS = 5 //constante

pontos: vetor[0..QUANTIDADE_DE_PONTOS-1] de REGISTRO Ponto //vetor do tipo de
//dados de Registro Ponto, com tamanho QUANTIDADE_DE_PONTOS

menor_distancia_ao_quadrado: INTEIRO
ponto_mais_proximo: INTEIRO

PARA i=0 ATÉ QUANTIDADE_DE_PONTOS-1 PASSO 1 FAÇA
  LER pontos[i].x
  LER pontos[i].y
FIM-PARA

menor_distancia_ao_quadrado = MAIOR_INTEIRO ❶
ponto_mais_proximo = 1 ❷

PARA i=1 ATÉ QUANTIDADE_DE_PONTOS-1 FAÇA
  distancia_ao_quadrado = (pontos[i].x-pontos[0].x)*(pontos[i].x-
            pontos[0].x)+(pontos[i].y-pontos[0].y)*  (pontos[i].y-pontos[0].y) ❸
  SE distancia_ao_quadrado < menor_distancia_ao_quadrado ENTÃO ❹
    ponto_mais_proximo = i ❺
    menor_distancia_ao_quadrado = distancia_ao_quadrado ❻
 FIM-SE
FIM-PARA
ESCREVER p[ponto_mais_proximo].x,p[ponto_mais_proximo].y
```

Observações:

❶❹❻ MAIOR_INTEIRO representa o maior número inteiro que podemos armazenar numa variável. Geralmente atribuímos **o maior** inteiro quando procuramos por **um menor** valor. No código, comparamos menor_distancia_ao_quadrado com distancia_ao_quadrado e salvamos o **menor** deles. Se executarmos isso sucessivamente, ao final, menor_distancia_ao_quadrado conterá o **menor** valor comparado.

❷❺ Esta variável irá guardar a posição do ponto mais próximo. Ela é atualizada, sempre que encontramos outro ponto com menor distância.

❸ Cálculo para encontrar a distância entre dois pontos. Na realidade, a distância entre os dois pontos seria a raiz quadrada de distancia_ao_quadrado. Mas não há diferença em comparar a distância ao quadrado. Sabemos, por exemplo, que a **raiz quadrada** de x é **menor do que a raiz quadrada** de y se x for **menor** do que y.

INT_MAX Para obtermos o maior inteiro em C, pode-se utilizar o comando INT_MAX da biblioteca <limits.h>

11. Fazer um programa em "C" que lê uma lista de 20 produtos e e preços e armazena-los em um *array* do tipo da estrutura abaixo. O programa deve, em seguida, ordenar o vetor em ordem alfabética de nome de produto e inflacionar os produtos cujos valores forem menores que 100 em 5%. Por fim a lista de produtos/preços deve ser impressa.

```
typedef struct {
char nome[80];
float preco;
} PROD;
```

OBS: usar uma rotina que recebe uma estrutura do tipo PROD com parâmetro e atualiza o preço, uma que lê os dados para a estrutura do tipo PROD e outra capaz de imprimir a estrutura. Para realizar essas três estruturas pode-se criar voids, por exemplo:
void atualiza (PROD itens) { ....}
void imprime (PROD itens) { ....}

DICA: A ordenação deve ser feita comparando o conteúdo da posição atual do vetor com conteúdo da posição seguinte deste vetor, caso o conteúdo seja maior deve-se, com uso de uma variável auxiliar, realizar a troca do conteúdo da posição seguinte do vetor para a posição atual do vetor.

12. Fazer um programa que lê um conjunto de informações (nome, endereço, CEP e data de nascimento) em um vetor de tamanho N. Deve-se ordenar de forma crescente por nome as informações no vetor e imprimir o vetor ordenado. Ou seja, deve-se solicitar N leituras do conjunto de informações e armazenar em um vetor, em seguida deve-se ordenar por nome as informações no vetor.

13. Fazer um programa que lê valores para uma matriz do tipo float de m linhas por n colunas e imprime a diferença entre a média dos elementos das colunas pares e a média dos elementos das linhas ímpares.

14. Foi realizada uma pesquisa de algumas características físicas da população de uma certa região, a qual coletaram os seguintes dados referentes a cada habitante:

- sexo (masculino e feminino);
- cor dos olhos (azuis, verdes ou castanhos);
- cor dos cabelos (louros, castanhos, pretos);
- idade.

Faça um programa para ler as informações de 50 entrevistados e salvar em um vetor de registro/struct. Ao final, determine e imprima:

a) A maior idade entre os entrevistados;
b) A quantidade de indivíduos do sexo feminino cuja idade está entre 18 e 35 anos, inclusive;
c) A quantidade de indivíduos que tenham olhos verdes e cabelos louros;

QUESTÕES SOBRE ALOCAÇÃO DINÂMICA:

15. Considerando o código em C produzido na Questão 10, altera-ló (em outro arquivo.c) de maneira a criar um menu de opções que permita: 1 - Ler cinco pontos de coordenadas cartesianas (x,y); 2 - Adicionar novos pontos cartesianos com a entrada da quantidade adicional de leituras; 3 - Alterar todos os pontos lidos, até o momento, no vetor; 4 - Obter o cálculo do ponto mais próximo do primeiro ponto lido (em relação a opção 1 do menu - mesmo que já tenham sido adicionados mais de cinco pontos, pela opção 2); 5 - Sair.

DICAS: Para serem possíveis as opções supracitadas deve-se alocar um vetor dinâmico inicialmente pelo comando malloc. Já para adicionar novos pontos cartesianos deve-se utilizar a realocação dinâmica do vetor aplicando o comando realloc. Sempre ao final do uso do vetor deve-se liberar a memória atribuindo a instrução free.
Também estruturar a solução para uso de funções e invoca-las no menu de opção. Por exemplo, para realizar a leitura dos cinco primeiros pontos de coordenadas (opção 1) e também para a leitura na alteração de todos os pontos do vetor (opção 3) poderíamos possuir a void abaixo:

```
void leituraPontos(struct Ponto coordenada[], int tam){ /*Passa por parâmetro um
                                    vetor do  tipo de dados struct Ponto e seu tamanho */
      int i;
      for(i=0; i < tamanho; i++){   printf("\nInforme as coordenadas do ponto(%i): ",i+1);
        scanf("%d %d",&coordenada[i].x,&coordenada[i].y);
      }
}
```

OBS: Deve-se definir a constante #define QUANTIDADE_DE_PONTOS 5 para atribuir o tamanho inicial do vetor a ser alocado dinamicamente.

16. Fazer um programa que solicita a leitura de 10 números inteiros. Armazenar os números em um vetor dinâmico. Após a leitura dos 10 números solicitar ao usuário se deseja ler mais valores, caso afirmativo, realocar mais espaços ao vetor. Imprimir os números correspondentes em hexa e octal.

Dica (identifique entre os comando abaixo quais devem ser utilizados):
Veja alguns dos especificadores de tipos de dados que podem ser lidos e impressos:
%c - Lê e imprime um carácter
%d - Lê e imprime um inteiro decimal
%l - Lê e imprime um inteiro decimal long
%f - Lê e imprime um número em ponto flutuante
%e - Lê e imprime um número em ponto flutuante em notação científica, por exemplo, 1000 igual a 1e+03, ou seja, 1 X 10^3
%o - Lê e imprime um número em octal
%s - Lê e imprime uma string
%x - Lê e imprime um número hexadecimal
%X Lê e imprime um número hexadecimal com letras, quando houver, em maiúsculas

QUESTÕES SOBRE PONTEIROS:

17. Consideremos as três funções do tipo void implementadas abaixo. Essas rotinas possuem o mesmo objetivo: incrementar em uma unidade o valor passado por parâmetro.

```
void soma(int k){

 k=k+1;

}

void adiciona(int x){

 x=x+1;

}

void incrementa(int *k){

 *k=*k+1;

}
```

Agora, na void main(), pode-se solicitar a leitura de uma variável inteira x, conforme segue:

```
int main (void){

int x;

printf("Digite o valor de X: ");

scanf("%d",&x);

soma(x);

printf("X= %d",x);

}
```

Responda:

a) Qual valor será impresso para x, caso na leitura seja digitado o valor 5? Por que imprime esse valor?

b) Se substituirmos na main() a chamada da void soma(x) pela chamada da outra void adiciona(x), o que será impresso em x, se digitado na leitura o valor 5? Por que imprime esse valor?

c) Em um último teste, se substituirmos na main() a chamada da void soma(x) pela chamada da outra void incrementa(&x), o que será impresso em x sendo lido o valor 5? Por que imprime esse valor?

Observe que entre essas diferenças está o princípio de funcionamento de ponteiros!

18. Mostre em uma tabela todos os passos para as variáveis x, y, *px, *py (teste de mesa) e identifique qual será a saída do programa em C, para os valores lidos (x = 3 e y = 4).

```
void proc(int *px, int *py) {

*py = (*py) * (*px);

*px = *px + 2;}

void main(void) {

int x, y;

scanf(“%d”,&x);

scanf(“%d”,&y);

proc(&x,&y);

printf(“x = %d, y = %d”, x, y);

}
```

19. Implemente um programa que declare um inteiro, um real e um char, e ponteiros para inteiro, real, e char. Associe as variáveis aos ponteiros (use &). Altere os valores de cada variável aplicando os ponteiros. Imprimir os valores das variáveis antes e depois das alterações.

20. Desenvolva uma rotina em C que contenha duas variáveis inteiras. Leia essas variáveis do teclado. Em seguida, compare seus endereços de memória e exiba o conteúdo do maior endereço, bem como o respectivo endereço em hexadecimal.

21. Escreva uma função do tipo void que dado um número real passado como parâmetro, retorne a parte inteira e a parte fracionária deste número. Escreva um programa que chama esta função. Assumir o protótipo:

void frac(float num, int *inteiro, float *frac);

22. Desenvolva um programa em C que realize a leitura de uma sequência de 10 números inteiros em um vetor estático e, que possua uma função com entradas por parâmetros e "devolva" os valores mínimo, máximo e a média dos inteiros lidos no vetor. O protótipo da função que realiza as operações deve ser:

void valores(int *min, int *max, float *med, int vetor[ ], int tam);

DICAS: Na void main() podem ser declaradas as variáveis inteiras mínimo (inicializada com o maior valor de um inteiro - INT_MAX da biblioteca <limits.h> -), máximo (inicializada como o menor valor de um inteiro - INT_MIN -) e tamanho_vetor. Uma a variável float media também é necessária. O resultado da invocação da função valores deverá "retornar" os valores de mínimo, máximo e média para as respectivas variáveis definidas na void main. Na implementação da void valores deverá ser percorrido o vetor passado por parâmetro e obtidos/calculados os valores desejados.

23. Implemente uma função que receba como parâmetro um vetor de números reais de tamanho N e retorne quantos números negativos há nesse vetor. A função deve obedecer ao protótipo:

int negativos(float *vet, int N);

24. Crie um programa com 4 vetores de tamanho 10 na função main. O primeiro vetor deverá conter os produtos vendidos. O segundo vetor armazena em cada posição a quantidade de produtos vendidos. O terceiro vetor armazena em cada posição o valor unitário de cada produto. O quarto vetor será utilizado para armazenar o valor total das vendas de cada produto. Assim, faça o que se pede:

a) Crie procedimentos (void) para imprimir os vetores na tela. Os procedimentos devem receber um ponteiro para o vetor a ser impresso e o tamanho do vetor.

b) Crie um procedimento (void) para calcular o valor total das vendas de cada produto e preencher o quarto vetor. O procedimento deverá receber ponteiros para os vetores e o tamanho dos vetores.

c) Crie um procedimento (void) que mostre qual produto foi mais vendido e qual o produto foi menos vendido.

d) Crie uma função (com retorno) que recebe um ponteiro para o quarto vetor, calcula o total arrecadado com todas as vendas e retorna este valor.

e) Crie a função main com os 4 vetores. O vetor produto, quantidade e valor unitário devem ser preenchidos com um conjunto de dados. Por fim, utilize as funções e procedimentos criados para imprimir os vetores, calcular o valor da venda de cada produto e o total das vendas.

25. Escreva uma função que aceita como parâmetro um vetor de inteiros com N valores, e determina o menor elemento do vetor e o número de vezes que este elemento ocorreu no vetor. Por exemplo, para um vetor com os seguintes elementos: 5, 2, 15, 3, 7, 2, 8, 6, 2, a função deve retorna para o programa que a chamou o valor 2 e o número 3 (indicando que o número 2 ocorreu 3 vezes). A função deve ser do tipo void.

26. Qual é a instrução que deve ser adicionada ao programa seguinte para que ele execute corretamente?

```
int main (void ) {

 int j, *pj;

 *pj = 3;

 printf("%d",*pj);

}
```