INSTITUTO FEDERAL DO ESPÍRITO SANTO

Programação em C

```
printf("I: Algoritmos \n
        II: Ferramentas da Linguagem C \n
        III: Introdução às Estruturas de Dados");
```

CAMPUS SERRA

Caderno de Exercícios e Questões

v1.1.0

Sumário

Cap. 1: INTRODUÇÃO

QUESTÕES PROPOSTAS

1) Explique, com suas próprias palavras, o que é um *Algoritmo*.

2) Algoritmos são necessariamente programas de computador? Explique.

3) Cite e explique algumas diferenças entre uma *linguagem de programação* e uma *pseudo-linguagem*.

4) Qual(is) a(s) diferença(s) entre uma linguagem *interpretada* e uma *compilada*?

5) Cite alguns motivos pelos quais é importante que todo programador conheça a linguagem C.

Cap. 2: A LINGUAGEM C

QUESTÕES PROPOSTAS

1) O que é uma linguagem de baixo nível? E alto nível? E o C, onde se enquadra?

2) C é uma linguagem didática? Por quê?

3) C é dito ser "*case sensitive*". Explique.

4) O que a função `main()`, presente em todo programa escrito em C, tem de especial?

5) Pra que servem as funções `printf()` e `scanf()`?

Cap. 3: Expressões e Comandos Básicos em C

Exercícios de Fixação

1) Escolha a opção que inclui somente nomes válidos para variáveis na linguagem C.

a. If, a_b_2, H789, _yes

b. i, j, int, obs

c. 9xy, a36, x*y, --j

d. 2_ou_1, \fim, *h, j

e. Nenhuma das opções anteriores

2) Qual das instruções abaixo está errada?

a. int i;

b. long float x;

c. long double y;

d. long ijk;

e. short int a;

f. unsigned b;

3) Em um compilador para um hardware de 16 bits uma variável `double` ocupa o mesmo espaço que _____ variáveis do tipo `char`.

a. Uma

b. Duas

c. Quatro

d. Oito

e. Dezesseis

4) Qual o trecho de programa que inicializa a variável z?

a. float z , z1=10.;

b. float z; z = 10;

c. float z = 10;

d. z = 10;

5) O trecho de programa a seguir é

```
main()
{
   char condicao;
   condicao = 'D';
   int i = 1;
}
```

a. Válido na linguagem C.

b. Não válido na linguagem C.

6) 0101 é uma constante _____ na linguagem C.

a. Binária

b. Hexadecimal

c. Inteira

d. Octal

e. Ponto Flutuante

7) Em C, `"t"` e `'t'` representam a mesma constante.

a. Verdadeiro.

b. Falso.

8) Diga o resultado das variáveis x, y e z depois da seguinte sequência de operações:

```
int x, y, z;
x = y = 10;
z = ++x;
x = -x;
y++;
x = x + y - (z--);
```

a. x = 11, y = 11, z = 11

b. x = -11, y = 11, z = 10

c. x = -10, y = 11, z = 10

d. x = -10, y = 10, z = 10

e. Nenhuma das opções anteriores.

9) Diga o resultado das variáveis x, y e z depois da seguinte sequência de operações:

```
int x,y;
int a = 14, b = 3;
float z;
x = a/b;
y = a%b;
z = y/x;
```

a. x = 4.66666, y = 2, z = 0.4286

b. x = 5, y =2, z= 0.4

c. x = 5, y = 2, z = 0

d. x = 4, y = 2, z = 0.5

e. x = 4, y =2, z = 0

f. Nenhuma das opções anteriores.

10) A operação lógica `(-5 || 0) && (3 >= 2) && (1 != 0) || (3 < 0)` é:

a. Verdadeira

b. Falsa

c. Inválida, pois sua sintaxe está errada.

d. Nem Verdadeira nem Falsa

e. Nenhuma das opções anteriores

11) Quais os valores de a, b e c após a execução do código abaixo?

```
int a = 10, b = 20, c;
c = a+++b;
```

a. a = 11, b = 20, c =30

b. a = 10 , b = 21, c = 31

c. a = 11, b = 20, c = 31

d. a = 10, b = 21, c = 30

e. Nenhuma das opções anteriores.

12) Qual o valor das variáveis v, x, y e z após a execução do seguinte trecho de código:

```
int v = 0, x = 1, y = 2, z = 3;
v += x+y;
x *= y = z + 1;
z %= v + v + v;
v += x += y += 2;
```

a. v=11, x=8, y=6, z=3

b. v=0, x=1, y=2, z=3

c. v=10, x=7, y=6, z=3

d. v=13, x=10, y=6, z=3

e. Nenhuma das opções anteriores.

Gabarito: 1-a; 2-b; 3-d; 4-c; 5-b; 6-d; 7-b; 8-b; 9-e; 10-a; 11-a; 12-d.

QUESTÕES PROPOSTAS

1) Determine o valor de cada uma das expressões abaixo. Dados: A=5, B=4, C=3 e D=6.

a. (A > C) && (C <= D)

b. (A+B) > 10 || (A + B) == (C + D)

c. (A>=C) && (D >= C)

d. (2*C) == D || (A+B) == 9

e. A*C/B > A*B*C

f. !0

g. !3 && 0

2) Determine o valor de cada uma das expressões abaixo. Dados: X=1, A=3, B=5, C=8 e D=7.

a. ! (X > 3)

b. (X < 1) && !(B > D)

c. !(D < 0) && (C > 5)

d. !((X > 3) || (C < 7))

e. (A > B) || (C > B)

f. (X >= 2)

g. (X < 1) && (B >= D)

h. (D < 0) || (C > 5)

i. !(D > 3) || !(B < 7)

j. (A > B) || !(C > B)

3) Determine o valor de cada uma das expressões abaixo. Dados: A=7, B=9, C=10 e D=0.

a. 3 + 5 >= A && A*B < B*C && A < C && !D

b. 2 + 5*3 > B*C && D

c. D || B > A && C > B || A > 5 && D

d. ((A*B – B*C > 54*C + B*10/3) && 258*A*C <= B*C/3 + 20) || !(A > B) [será necessária tanta conta?]

e. D || B > A && C > B || A > 5 && !D

f. D || ? || B > A && !D [**?** significando "*don't care*" ou "não importa"]

4) Faça o algoritmo para calcular a área e o perímetro de uma sala, sendo que as dimensões (comprimento e largura) são fornecidos pelo usuário. Apresente a área e o perímetro depois de calculados.

5) Faça o algoritmo para calcular quanto será pago por um produto, sendo que o preço do produto e o desconto são fornecidos pelo usuário. Apresentar o valor a ser pago pelo produto.

6) Faça o algoritmo para calcular o valor a ser pago pelo período de estacionamento do automóvel. O usuário entra com os seguintes dados: hora e minuto de entrada, hora e minuto de saída. Sabe-se que este estacionamento cobra R$ 4,00 por hora, mas calcula as frações de hora também. Por exemplo, se a pessoa ficar 1 hora e quinze minutos, pagará R$ 5,00 (R$ 4,00 pela hora e R$ 1,00 pelos quinze minutos).

7) Faça o algoritmo que receba dois valores inteiros nas variáveis A e B. O programa deve trocar os valores entre as variáveis (ou seja, ao término do programa a variável A deve ter o valor inicial de B e vice-versa). Apresentar as duas variáveis o final.

8) Faça um algoritmo que, a partir de um valor inteiro positivo fornecido pelo usuário, expresse-o em valores relativos. Ex: 123 = 1 centena, 2 dezenas e 3 unidades.

9) Escreva um algoritmo para ler o número total de eleitores de um município, o número de votos brancos, nulos e válidos. Calcular e escrever o percentual que cada um representa em relação ao total de eleitores.

10) Escreva um algoritmo para ler uma temperatura em graus Fahrenheit, calcular e escrever o valor correspondente em graus Celsius (baseado na fórmula abaixo):

$$\frac{C}{5} = \frac{F - 32}{9}$$

Observação: Para testar se a sua resposta está correta saiba que 100° C = 212° F

11) O custo de um carro novo ao consumidor é a soma do custo de fábrica com a porcentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que o percentual do distribuidor seja de 28% e os impostos de 45%, escrever um algoritmo para ler o custo de fábrica de um carro, calcular e escrever o custo final ao consumidor.

Cap. 4: Comandos de Controle do Programa

Exercícios de Fixação

1) `if(num)...;` é equivalente a `if(num!=0)...;`

a. Verdadeiro.

b. Falso.

2) Qual o valor de x após a seguinte sequência de comandos:

```
a = 10;
b = 20;
x = 0;
x = (b > a) ? b : a;
```

a. 0

b. 2

c. 10

d. 20

e. 40

f. Nenhuma das opções anteriores.

3) Os trechos de programa a seguir são equivalentes entre si, sob o ponto de vista do que é impresso:

```
for (i = 0 ; i < 10; i++) printf("%d", i);
```

e

```
for (i = 0 ; i < 10; ++i) printf("%d", i);
```

a. Verdadeiro.

b. Falso.

4) O trecho de programa a seguir é

```
switch(num)
{
   case 1;
      printf("O numero eh 1");
      break;
   case 2;
      printf("O numero eh 2");
      break;
   default;
      printf("O numero eh diferente de 1 e 2");
      break;
}
```

a. Válido na linguagem C.

b. Não válido na linguagem C.

5) Sendo `num` uma variável inteira, o que imprime o trecho de código a seguir?

```
num = 1;
switch(num)
{
case 1:
   printf("O numero eh 1 ");
case 2:
   printf("O numero eh 2 ");
default:
   printf("O numero eh diferente de 1 e 2");
}
```

a. O numero eh 1
b. O numero eh 2
c. O numero eh diferente de 1 e 2
d. O numero eh 1 O numero eh 2
e. O numero eh 1 O numero eh 2 O numero eh diferente de 1 e 2

6) Os dois blocos de código a seguir produzem o mesmo resultado:

```
for(i = 0 ; i < 3; i++)
   for(j = 0 ; j < 3; j++)
      printf("i+j = %d \n", i+j);
```

e

```
for( i = 0 , j = 0; i < 3; i++)
   for ( ; j < 3; j++)
      printf("i+j = %d \n", i+j);
```

a. Verdadeiro.
b. Falso.

7) Qual a saída produzida pelo extrato de código a seguir:

```
int x;
for(x = 35; x > 0; x/=3)
   printf("%d  ", x);
```

a. 35 11 3 1
b. 11 3 1
c. 11 3 1 0
d. 35 11 3
e. Nenhuma das opções anteriores.

8) Os extratos de código a seguir são equivalentes entre si:

```
int x = 10;
while (--x > 9)
{
   printf("%d", x);
}
```

e

```c
int x = 10;
do {
   printf("%d", x);
} while(--x > 9);
```

a. Verdadeiro.

b. Falso.

9) Sendo i declarado e inicializado como:

```c
int i = 0;
```

os seguintes extratos de código:

```c
while (i = 5)
{
   printf("%d %d %d \n", i, i+2, i+3);
   i = 0;
}
```

e

```c
if (i = 5) printf("%d %d %d \n", i, i+2, i+3);
```

a. São idênticos sob o ponto de vista do que imprimem na tela.

b. Não imprimem nada na tela.

c. Têm sintaxe errada.

d. Um deles imprime 5, 7 e 9 uma única vez e o outro entra em loop, imprimindo estes valores indefinidamente.

e. Nenhuma das opções anteriores.

10) A estrutura do switch abaixo é:

```c
switch(t)
{
case t < 10:
   printf("Hoje ta fazendo muito frio");
   break;
case t < 25:
   printf("A temperatura esta agradavel");
   break;
default:
   printf("Hoje ta quente pra caramba");
}
```

a. Válida na linguagem C.

b. Não válida na linguagem C.

11) O laço for a seguir

```c
int i;
for (i = 0; i <= 5; i++, printf("%d ", i));
```

a. Imprime 0 1 2 3 4 5.

b. Não funciona, pois tem sintaxe errada.

c. Imprime 1 2 3 4 5 6.

d. Imprime 1 2 3 4 5.

e. Nenhuma das opções anteriores.

Gabarito: 1-a; 2-d; 3-a; 4-b; 5-e; 6-b; 7-a; 8-b; 9-d; 10-b; 11-c.

QUESTÕES PROPOSTAS

COMANDOS DE SELEÇÃO

1) Construa um algoritmo que leia o valor de uma conta de luz e, caso o valor seja maior que R$ 50,00 apresente a mensagem: “Você está gastando muito”. Caso contrário não exiba mensagem nenhuma.

2) Construa um algoritmo que leia o preço de um produto e apresente a mensagem: “Em promoção”, caso o preço seja maior ou igual a R$ 50,00 e menor ou igual a R$ 100,00. Caso contrário, deve apresentar a mensagem: “Preço Normal”.

3) Elabore um algoritmo que leia três números e os imprima em ordem crescente. Desconsidere números iguais. Após desenvolvido e testado, pergunte-se: “É esta a forma mais *eficiente* de ordenar 3 números usando um comando de seleção?”

4) Construa um algoritmo que calcule o novo salário de um funcionário. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário seja menor que 500. Se o salário for maior ou igual a 500, mas menor ou igual a 1000, o reajuste deve ser de 10%. Caso o salário seja maior que 1000, o reajuste deve ser de 5%.

5) Faça um algoritmo que verifique se um funcionário já pode se aposentar, a partir da leitura do seu ano de nascimento e do tempo de serviço. Ele poderá se aposentar se *pelo menos uma* das condições abaixo for verdadeira:
   - Ter no mínimo 65 anos de idade.
   - Ter trabalhado, no mínimo, 30 anos.
   - Ter no mínimo 60 anos de idade e ter trabalhado no mínimo 25 anos

6) Construa um algoritmo que, tendo como dados de entrada a altura (H) e o sexo de uma pessoa calcule e apresente seu peso ideal utilizando as seguintes fórmulas:

   Para homens: Peso ideal = (72,7 * H) – 58

   Para mulheres: Peso ideal = (62,1 * H) – 44,7

7) Construa um algoritmo que, dados os comprimentos dos três lados (A, B e C) de um triângulo, verifique o tipo de triângulo formado. Apresentar qual é o tipo. Sabe-se que:
   - Triângulo do tipo Equilátero – possui os três lados iguais
   - Triangulo do tipo Isósceles – possui dois lados iguais
   - Triângulo do tipo Escaleno – possui os três lados diferentes

   OBS: Quaisquer 3 segmentos formam um triângulo? Verifique e valide, portanto, a entrada.

8) Construa um algoritmo que, a partir do peso e altura do usuário, calcule o IMC (Índice de Massa Corporal) e apresente a "categoria" na qual o usuário se encontra.

IMC = Peso / Altura² (kg/m²)

IMC	Classificação
< 18,5	Magreza
18,5 – 24,9	Saudável
25,0 – 29,9	Sobrepeso
30,0 – 34,9	Obesidade Grau I
35,0 – 39,9	Obesidade Grau II
≥ 40,0	Obesidade Grau III

9) Construa um algoritmo que tendo como dados de entrada o preço base de um produto e seu código de origem, apresente o preço final (considerando o frete) e a sua procedência, de acordo com a tabela abaixo:

Código de Origem	Procedência	Valor do Frete
1	Sul	R$ 15,00
2	Sudeste	R$ 5,00
3	Centro-Oeste	R$ 20,00
4	Norte	R$ 25,00
5	Nordeste	R$ 20,00

10) Construa um algoritmo que determine quanto será gasto para encher o tanque de um carro, sabendo-se que o preço da gasolina é de R$ 2,80 e o preço do álcool é de R$ 1,95. O usuário fornecerá os seguintes dados: Tipo de carro (G – gasolina ou A – álcool) e Capacidade do tanque, em litros.

11) Construa um algoritmo que leia as informações de: horas trabalhadas, valor da hora trabalhada. Calcule e apresente o salário líquido do empregado, baseado nas tabelas abaixo.

OBS: Salário Líquido = Salário Bruto – INSS – Imposto de Renda

- Salário Bruto = Horas trabalhadas * Valor da hora trabalhada
- INSS = 11% do salário bruto
- Imposto de Renda = Alíquota * (Salário Bruto – INSS) – Valor a Deduzir

Salário Bruto – INSS	Alíquota	Valor a Deduzir
Até R$ 1.499,15	(Isento)	(Isento)
De R$ 1.499,16 até R$ 2.246,75	7,5%	R$ 112,43
De R$ 2.246,76 até R$ 2.995,70	15%	R$ 280,94
De R$ 2.995,71 até R$ 3.743,19	22,5%	R$ 505,62
Acima de R$ 3.743,20	27,5%	R$ 692,78

12) Construa um algoritmo que calcule e apresente quanto deve ser pago por um produto considerando a leitura do preço de etiqueta e o código da condição de pagamento. Utilize para os cálculos a tabela de condições de pagamento a seguir:

Código da condição de pagamento	Condição de pagamento
1	À vista em dinheiro ou cheque, com 10% de desconto
2	À vista com cartão de crédito, com 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço de etiqueta com acréscimo de 10%

13) Construa um algoritmo que, a partir da data atual (dia, mês e ano atual) e a data de nascimento de uma pessoa (dia, mês e ano) apresente sua idade real, em anos.

14) Construa um algoritmo que leia o código de um determinado produto e mostre a sua classificação, utilizando a seguinte tabela:

Código do Produto	Classificação
1	Alimento não perecível
2, 3 ou 4	Alimento perecível
5 ou 6	Vestuário
7	Higiene Pessoal
8 ou 9	Limpeza e Utensílios Domésticos
Qualquer outro código	Inválido

COMANDOS DE ITERAÇÃO

15) Construa um programa que calcule N! (fatorial de N), sendo que o valor de N (inteiro) é fornecido pelo usuário. Sabe-se que:

N! = 1 x 2 x 3 x 4 x .... x N

OBS: 0! = 1 (fatorial do número zero é igual a 1 por definição). Além disso, não deve ser permitido que seja calculado o fatorial de número negativo, pois isso não existe.

16) Um número é primo se e somente se ele é divisível apenas por 1 e por ele mesmo. Elabore um algoritmo que, dado um número N >= 2, diga se ele é primo ou não.

17) Faça um algoritmo que calcule a soma dos números primos entre 1 e 100.

18) Faça um algoritmo que imprima os 20 primeiros termos da série abaixo (Série de Fibonacci):

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

19) Dada a definição de MDC: “Dados dois números inteiros **a** e **b** não nulos, define-se o máximo divisor comum como sendo o maior inteiro que divide simultaneamente **a** e **b**”. Faça um algoritmo que leia dois números e, a partir deles, descubra o máximo divisor comum.

20) Em um cinema, certo dia, cada espectador respondeu a um questionário, que perguntava a sua idade e a opinião em relação ao filme, seguindo os seguintes critérios:

Opinião	Significado
A	Ótimo
B	Bom
C	Regular
D	Ruim
E	Péssimo

A entrada de dados sobre a opinião deve ser validada.

O final da pesquisa será indicado quando a idade do usuário for informada como negativa (idade inexistente).

Construa um programa que, lendo esses dados, calcule e apresente:

- Quantidade de pessoas que respondeu a pesquisa
- Média de idade das pessoas que responderam a pesquisa
- Porcentagem de cada uma das respostas

21) Faça um algoritmo que calcule a soma dos N primeiros termos da série abaixo, onde N é um valor inteiro positivo fornecido pelo usuário:

$$S = -1 + \frac{2}{4} - \frac{3}{9} - \frac{5}{25} + \frac{6}{36} - \frac{7}{49} - \frac{9}{81} + \cdots$$

22) Faça um algoritmo para ler números inteiros e positivos. Encerrar o processamento quando for digitado -1. Calcular e mostrar:

a. A média geral dos números lidos.
b. A média dos números pares maiores que 50 e menores que 100.
c. Maior e o menor número.
d. A soma dos números menores e igual a 40 e maiores e iguais a 200.

23) Elabore um algoritmo que exiba na tela todos os fatores primos resultantes da decomposição de um número inteiro e positivo. Ex: 54 = 2 x 3 x 3 x 3 x 1

24) Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos, indagou à rainha sobre o pagamento, se poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez (que possui 64 casas), de tal forma que o primeiro quadro deveria conter apenas um grão e os quadros subsequentes, o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Faça um algoritmo para calcular o número de grãos que o monge esperava receber.

25) Fulano tem 1,50 metro e cresce 2 centímetros por ano, enquanto Ciclano tem 1,10 metro e cresce 3 centímetros por ano. Construa um algoritmo que calcule e imprima quantos anos serão necessários para que Ciclano seja maior que Fulano.

26) Em uma eleição presidencial, existem quatro candidatos. Os votos são informados através de código. Os dados utilizados para a escrutinagem obedecem à seguinte codificação:

- 1, 2, 3, 4 = voto para os respectivos candidatos;
- 5 = voto nulo;
- 6 = voto em branco;

Elabore um algoritmo que calcule e escreva:

- Total e percentual de votos válidos para cada candidato;
- Total de votos nulos;
- Total de votos em branco;
- Percentual dos votos em branco e nulos sobre o total;
- Situação do candidato vencedor sobre os outros três, no caso, se ele obteve ou não mais votos que os demais somados;

Como finalizador do conjunto de votos, tem-se o valor 0.

27) Construa um algoritmo que leia um conjunto de dados contendo altura e sexo (masculino e feminino) de 50 pessoas e depois calcule e escreva:

- A maior e a menor altura do grupo;
- A média de altura das mulheres;
- O número de homens e a diferença percentual entre estes e as mulheres.

28) Faça um algoritmo para ler dois valores inteiros. O primeiro representa o primeiro elemento de uma progressão aritmética, e o segundo a razão. Calcule e mostre os 10 elementos seguintes.

29) Faça um algoritmo para ler um número menor que 1000 e calcular e mostrar todos os números ímpares maiores que o número lido e menores que 1000.

30) Faça um algoritmo que calcule a soma dos N primeiros termos da série abaixo, onde N é um valor inteiro positivo fornecido pelo usuário:

$$S = 1000 - \frac{997}{2} + \frac{994}{3} - \frac{991}{4} + \cdots$$

31) Solicite ao usuário um número inteiro N positivo maior que 0, obrigando o usuário a fornecer um número nessas condições. Em seguida, exiba todos os inversos dos números partindo de –N até N. Use um comando de desvio apropriado para evitar a divisão por zero. Ex: N=3→ 1/-3, 1/-2, 1/-1, 1/1, 1/2, 1/3. (aqui representados na forma de fração, mas no algoritmo devem estar em ponto flutuante).

Cap. 5: VETORES, MATRIZES E STRINGS

EXERCÍCIOS DE FIXAÇÃO

1) Seja um vetor declarado por:

```
int vet[10];
```

Qual elemento deste vetor é acessado quando se escreve `vet[2]`?

a. Primeiro elemento.
b. Segundo elemento.
c. Terceiro elemento.
d. Quarto elemento.
e. Nenhuma das opções anteriores.

2) Se declararmos um vetor como:

```
int vet[30];
```

a instrução abaixo acessa corretamente os elementos deste vetor?

```
for (j = 0; j <= 30; j++)
   vet[j] = j*j;
```

a. Sim.
b. Não.

3) Seja a matriz `matrx` declarada e inicializada por:

```
int matrx[][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

O que conterá o elemento `matrx[1][2]`?

a. 2
b. 5
c. 6
d. 7
e. Nenhuma das opções anteriores.

4) Se uma string for declarada como mostrada abaixo. O número máximo de caracteres que poderão ser lidos e armazenados nela é:

```
char str[20];
```

a. 18
b. 19
c. 20

d. 21

5) Qual função pode ser usada para determinar o comprimento de uma string?

a. gets

b. strcpy

c. strcat

d. strlen

e. strcmp

Gabarito: 1-c; 2-b; 3-d; 4-b; 5-d.

QUESTÕES PROPOSTAS

VETORES E MATRIZES

1) Preencher um vetor B de 10 elementos com 1 se o índice do elemento for ímpar e com 0 se for par. Escrever o vetor B após o seu total preenchimento.

2) Ler um vetor X de 10 elementos inteiros e positivos. Criar um vetor Y da seguinte forma: os elementos de Y com índice par receberão os respectivos elementos de X divididos por 2; os elementos com índice ímpar receberão os respectivos elementos de X multiplicados por 3. Escrever o vetor X e o vetor Y.

3) Ler um vetor W de 10 elementos, depois ler um valor V. Contar e escrever quantas vezes o valor V ocorre no vetor W e escrever também em que posições (índices) do vetor W o valor V aparece. Caso o valor V não ocorra nenhuma vez no vetor W, escrever uma mensagem informando isto.

4) Dado dois vetores, A e B, contendo valores inteiros não repetidos (num mesmo vetor) faça um programa em C que imprima todos os elementos comuns aos dois vetores. Informe também a quantidade de elementos comuns encontrados.

5) Armazenar em um vetor vinte números digitados pelo usuário. Em um segundo vetor armazenar a soma dos elementos do primeiro vetor da seguinte maneira: valor da posição 1 com valor da posição 20, valor da posição 2 com valor da posição 19, etc. Apresentar o resultado deste novo vetor.

6) A partir de um vetor inteiros de tamanho N (ex: N = 12), lido pelo teclado ou inserido diretamente no código, verifique se os elementos estão em progressão aritmética.

7) Crie um vetor de 10 elementos e solicite ao usuário que digite cada um dos elementos, um a um. Considere que os elementos do vetor estão dispostos de forma circular, determine uma rotação de k elementos, sem o uso de um vetor auxiliar.

   Exemplo:

   A = 1, 3, 5, 4, 7, 9, 12, 6, 20 e k = 3

   A = 12, 6, 20, 1, 3, 5, 4, 7, 9

8) São dados dois vetores com 10 posições cada. Cada um dos vetores tem seus elementos em ordem não decrescente (ou seja, é crescente ou constante). Gere um terceiro vetor que intercala os elementos de cada um dos vetores, em ordem não decrescente.

Exemplo:

A = 1, 5, 6, 9, 15, 15, 18 ...

B = 2, 3, 7, 10, 18, 20, 25 ...

AB = 1, 2, 3, 5, 6, 7, 9, 10, 15, 15, 18, 20, 25 ...

9) Altere o algoritmo da questão anterior para evitar repetições no vetor resultante.

10) Declare uma matriz 5 x 5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Escreva ao final a matriz obtida.

11) Leia uma matriz 10 x 10 e escreva a localização (linha e a coluna) do maior valor.

12) Leia duas matrizes 4 x 4 e escreva uma terceira com os maiores elementos entre as primeiras (em suas respectivas posições).

13) Dada uma matriz A (n x m) imprimir o número de linhas e o número de colunas nulas da matriz.

Exemplo:

```
0 0 0 0
1 0 2 2
4 0 5 6
0 0 0 0
```

tem duas linhas nulas e uma coluna nulas.

14) Dada uma matriz M (10, 20), preenchê-la por leitura e imprimir:
   a) O maior elemento de cada linha da matriz;
   b) A média dos elementos de cada coluna;
   c) O produto de todos os elementos diferentes de zero;
   d) Quantos elementos são negativos;
   e) Posição ocupada (linha-coluna) por um elemento cujo valor será lido pelo programa.

15) A turma do primeiro período de Engenharia de Automação do IFES que possui *22 alunos* será avaliada na disciplina de Algoritmos e Estrutura de Dados usando *3 provas*, cujas notas vão de 0 a 100, incluindo valores decimais. Sabendo disso, elabore um algoritmo que:
   a) Exiba as 3 notas de cada aluno (uma em cada coluna) e outras duas colunas extras: Média Parcial e Situação (se MédiaParcial >= 60 → Aprovado, caso contrário, está de Prova Final)
   b) Calcule a média e o desvio padrão[1] da turma em cada uma das provas.
   c) Exiba quantos alunos estão Aprovados e quantos estão de Prova Final.

[1] O desvio padrão é uma medida de dispersão de um conjunto de valores em torno da média, e é dado pela seguinte fórmula:

$$\sigma = \sqrt{\frac{\sum(X_i - \overline{X})^2}{n-1}}$$

onde:

X_i = Valor de cada elemento individual (X_1, X_2, X_3, ... , X_n)

$\overline{X}$ = Média aritmética dos valores de X_i

STRINGS

16) Solicite ao usuário duas strings. A seguir, posicione as strings de maneira ordenada, usando os símbolos >, < ou =. Ex:

```
Digite a string 1: trabalho
Digite a string 2: sucesso
Resultado: sucesso > trabalho
```

17) Fazer um programa em C que leia uma frase e imprima somente as vogais.

18) Usando um vetor de strings (matriz de `char`) e um vetor de `char` adicional, solicite o nome e o sexo ('M' ou 'F') de 10 pessoas. Em seguida solicite ao usuário se ele deseja visualizar os nomes do sexo masculino ou feminino. Exiba-os.

19) Uma palavra é denominada um palíndromo se for invertida e a leitura da mesma permanecer sem nenhuma alteração. Algumas palavras que são palíndromos são: aba, radar, reter, rever, rir, rotor, dentre outras. Construir um programa que detecte se uma palavra (string) digitada pelo usuário é ou não um palíndromo.

20) Construir um programa que conta e imprime o número de ocorrências de uma letra, fornecida pelo usuário, em uma dada string, também digitada pelo usuário.

21) Construir um programa que pede para o usuário:
    - Uma string s,
    - Um caractere ch1,
    - Um caractere ch2.

    O programa deve substituir todas as ocorrências do caractere ch1 em s pelo caractere ch2.

22) Fazer um programa para concatenar duas strings lidas do usuários. A segunda string deve ser concatenada na primeira. (Não use funções prontas de contatenação)

    Exemplo:
    Digitado: string1: sol string2: lua
    Após concatenar: string1: sollua string2: lua

23) Fazer um programa semelhante ao anterior, mas agora as duas strings devem ser concatenadas e atribuídas a uma terceira string. Exibir o tamanho de todas elas após a concatenação. (Não use funções prontas de contatenação)

    Exemplo:
    Digitado: string1: sol string2: lua string3: sollua
    Tamanho: 3, 3, 6

24) Fazer um programa para ler uma string e um caractere. Sempre que o caractere lido aparecer na frase ele deve ser substituído por asterisco.

    Exemplo:
    Frase: o dia esta nublado
    Caractere: d
    Resultado: o *ia esta nubla*o

25) Crie 2 strings, uma com capacidade 100 e outra com 30 caracteres. Na string menor, solicite ao usuário o seu nome. Uma vez capturado o nome, copie-o para a string maior. Em seguida, usando esta mesma string menor, solicite cada um dos seus sobrenomes e vá concatenando a string maior. Ao final, exiba a string maior e a quantidade de caracteres nela contidos (seu comprimento).

INTRODUÇÃO A PESQUISA E ORDENAÇÃO

26) Leia um vetor de 20 posições e em seguida um valor X qualquer. Seu programa devera fazer uma busca do valor de X no vetor lido e informar a posição em que foi encontrado ou se não foi encontrado.

27) Solicite ao usuário 10 números inteiros quaisquer e armazene-os em um vetor. Com base no vetor, encontre:
    a) Maior e o menor número, sem usar ordenação.
    b) Maior e o menor número, usando algum mecanismo de ordenação.

28) O *Bubble Sort* e o *Selection Sort* são dois algoritmos de ordenação considerados dos mais simples. Na apostila é possível encontrar ambos implementados. Explique com um exemplo numérico o princípio de funcionamento de cada um deles.

Cap. 6: INTRODUÇÃO À MODULARIZAÇÃO: SUBPROGRAMAS

QUESTÕES PROPOSTAS

TEORIA

1) Uma boa especificação é uma parte importante na etapa de planejamento de um sistema. Cite algumas razões para que isso seja feito.

2) Que ligação existe entre o planejamento e a manutenção de um sistema?

3) É importante pensar bem em como o seu sistema se comportará em caso de mau funcionamento? Por quê? Cite exemplos que apoiem sua resposta.

4) Uma vez desenvolvida uma solução para um problema, isto significa que o trabalho do projetista do software chegou ao fim? Por quê?

5) Cite alguns benefícios de se desenvolver um sistema em módulos. Apoie sua resposta com exemplos práticos.

6) Explique o importante princípio da "Caixa Preta", em desenvolvimento de sistemas.

7) Cite os 4 critérios úteis na modularização. Apoie-os com exemplos *diferentes* da apostila.

8) Explique o que ocorre quando chamamos uma função, no que diz respeito ao que acontece na passagem dos parâmetros.

9) O que significa uma função ter um retorno do tipo `void`?

10) O que é um bloco de código? Qual a sua relação com a declaração de variáveis?

11) Conceitue variáveis locais, variáveis globais e parâmetros formais.

12) Variáveis globais, apesar de úteis, seu uso exige cautela. Por quê? (Cite dois motivos).

PRÁTICOS

1) Escreva uma função que converta uma temperatura em graus Fahrenheit para graus Celsius, baseado na fórmula abaixo:

$$\frac{C}{5} = \frac{F - 32}{9}$$

2) Elabore um algoritmo que exiba na tela todos os fatores primos resultantes da decomposição de um número inteiro e positivo. Ex: 54 = 2 x 3 x 3 x 3 x 1. Porém, sua abordagem deverá ser modularizada em funções. Para tanto, você deve:
   a. Criar uma função `EhNumeroPrimo` que, a partir de um número inteiro passado como parâmetro, retorne um valor indicando se o número é ou não primo.

b. Crie uma função `ProximoNumeroPrimo` que, a partir de um número inteiro passado como parâmetro, retorne o próximo número primo. Esta função faz uso da função acima.

c. No algoritmo principal, elabore a lógica pedida no enunciado fazendo uso das funções criadas.

3) Elabore um programa chamado *Utilitários Matemáticos*. Na função principal `main()` deverá ter um menu com a estrutura exibida abaixo e se encarregará apenas de parte da interface com o usuário. Ela deverá guiar o usuário através dos menus/submenus (implementados com o `switch`) até que por fim chegue ao utilitário (Ex: Cálculo de Áreas > Triângulo). Cada utilitário deverá ser implementado usando uma função que se encarregará apenas de pegar todos os dados necessários aos cálculos, que será passado para outra(s) *função(ões)* que fará(ão) os cálculos necessários. Nas funções que efetuam os cálculos propriamente ditos, não deverá haver quaisquer comandos de escrita/leitura de dados, cabendo a elas apenas o processamento lógico/numérico. Não use variáveis globais.

Estrutura/Funcionalidades:

- Análise Combinatória
  - Permutação
  - Combinação
  - Arranjo
- Áreas
  - Retângulo
  - Círculo
  - Triangulo
- Funções (devido a impossibilidade de se retornar dois valores, exiba as raízes calculadas na própria função)
  - Raízes de uma equação de 2º Grau
- Juros
  - Simples
  - Compostos
- Volumes
  - Cubo
  - Esfera
  - Cone
- Progressões
  - Aritmética
    - N-ézimo termo
    - Soma dos termos
  - Geométrica
    - N-ézimo termo
    - Soma dos termos

Cap. 7: Criação, Compilação e Execução de Programas em C

Questões Propostas

1) O que vem a ser a Biblioteca Padrão do C e qual a sua importância na eficiência do trabalho do programador?

2) O que é *linkedição*?

3) Explique, passo a passo e de maneira resumida, o processo de criação de um programa em C.

4) (a) Cite alguns dos papeis do pré-processador. (b) Há alguma relação entre as funções desempenhadas por ele e pelo compilador C, propriamente dito? Em outras palavras, estão esses dois elementos de alguma maneira interdependentes?

5) Considerando o mapa de memória do C, responda:
   a. Quais são alguns dos usos da região de *Pilha*?
   b. Variáveis *Globais* e Variáveis *Locais* são armazenadas da mesma maneira? Tente explicar o motivo pelo qual as variáveis locais são armazenadas assim.
   c. Para que serve a região conhecida como *Heap*?
   d. Quando o programa é carregado na memória, dados e instruções se misturam?

6) Qual(is) a(s) utilidade(s) da diretivas:
   a. #include
   b. #define

Cap. 8: PONTEIROS

EXERCÍCIOS DE FIXAÇÃO

1) Qual das instruções abaixo é correta para declarar um ponteiro para inteiro?

a. *int pti;

b. *pti;

c. &i;

d. int_pti pti;

e. int *pti;

2) Seja a seguinte sequência de instruções em um programa C:

```
int *pti;
int i = 10;
pti = &i;
```

Qual afirmativa é falsa?

a. pti armazena o endereço de i.

b. *pti é igual a 10.

c. ao se executar *pti = 20; i passará a ter o valor 20.

d. ao se alterar o valor de i, *pti será modificado.

e. pti é igual a 10.

3) Se i e j são variáveis inteiras e pi e pj são ponteiros para inteiro, qual atribuição é ilegal?

a. pi = &i;

b. *pj = &j;

c. pj = &*&j;

d. i = *&*&j;

e. i = (*pi)+++*q;

4) Seja a seguinte sequência de instruções em um programa C:

```
int *pti;
int veti[] = { 10, 7, 2, 6, 3};
pti = veti;
```

Qual afirmativa é falsa?

a. *pti é igual a 10.

b. *(pti+2) é igual a 2.

c. pti[4] é igual a 3.

d. pti[1] é igual a 10.

e. *(veti+3) é igual a 6.

5) Na sequência de instruções abaixo:

```
float f;
float *pf;
pf = &f;
scanf("%f", pf);
```

a. Efetuamos a leitura de f.
b. Não efetuamos a leitura de f.
c. Temos um erro de sintaxe.
d. Deveríamos estar usando &pf no scanf().
e. Nenhuma das opções anteriores.

6) Seja a declaração:

```
int matr[][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Qual afirmativa é falsa?

a. **matr é igual a 1.
b. *(*(matr+1)+2) é igual a 7.
c. *(matr[2]+3) é igual a 12.
d. (*(matr+2))[2] é igual a 11.
e. *((*matr)+1) é igual a 5.

Gabarito: 1-e; 2-e; 3-b; 4-d; 5-a; 6-e.

EXERCÍCIOS RESPONDIDOS E COMENTADOS

ENUNCIADOS

1) Explique a diferença entre as expressões abaixo, considerando que `p` é um ponteiro:
`p++; (*p)++; *(p++);`

2) O que quer dizer `*(p+10);` considerando que `p` é um ponteiro?

3) Explique o que você entendeu da comparação entre ponteiros.

4) Qual o valor de `y` no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um `/* comentário */` em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução.

```
void main() {
   int y, *p, x;
   y = 0;
   p = &y;
   x = *p;
   x = 4;
   (*p)++;
   x--;
   (*p) += x;
   printf("y = %d\n", y);
}
```

5) Fizemos a função StrCpy(). Faça uma função StrLen() e StrCat() que funcionem como as funções strlen() e strcat() de string.h, respectivamente.

6) Escreva a função int strend(char *s, char *t) que retorna 1 (um) se a cadeia de caracteres 't' ocorrer no final da cadeia 's', e 0 (zero) caso contrário.

7) Verifique o programa abaixo. Encontre o seu erro e corrija-o para que escreva o numero 10 na tela.

```c
#include <stdio.h>
int main()
{
   int x, *p, **q;
   p = &x;
   q = &p;
   x = 10;
   printf("\n%d\n", &q);
   return(0);
}
```

8) Escreva um programa que declare uma matriz 100x100 de inteiros. Você deve inicializar a matriz com zeros usando ponteiros. Preencha depois a matriz com os números de 1 a 10.000 usando ponteiros.

9) Aprendemos, pelo curso, que o valor de uma variável ou expressão do tipo vetor é o endereço do elemento zero do vetor. Seja `a[]` um vetor qualquer, independente de tipo e tamanho, e `pa` um ponteiro para o mesmo tipo de `a[]`. Responda V ou F, justificando:

( ) Após a atribuição `pa=&a[0];` `pa` e `a` possuem valores idênticos, isto é, apontam para o mesmo endereço.

( ) A atribuição `pa=&a[0];` pode ser escrita como `pa=a;`

( ) `a[i]` pode ser escrito como `*(a+i)`

( ) `&a[i]` e `a+i` são idênticos

( ) `a+i` é o endereço do i-ésimo elemento do vetor `a`

( ) `pa[i]` é idêntico a `*(pa+i)`

( ) `pa=a` é uma operação válida

( ) `pa++` é uma operação válida

( ) `a=pa` é uma operação válida

( ) `a++` é uma operação válida

10) O que está errado com os programas abaixo? Descubra e indique a solução para consertá-los. Execute-o no computador para ver se o erro foi resolvido.

a) Programa 1

```c
/* Esse programa está errado */
void main()
{
   int x, *p;
   x = 10;
   *p = x;
}
```

OBS: por se tratar de um programa muito pequeno, talvez a execução deste não mostrará de imediato o erro. Mas ele existe e é sério.

b) Programa 2

```
/* Esse programa está errado */
void main()
{
   int x, *p;
   x = 10;
   p = x;
   printf ("%d", *p);
}
```

RESPOSTAS E COMENTÁRIOS

1) Solução

`p++`: incrementa o ponteiro, ou seja, o endereço. Após esta instrução, o ponteiro `p` passará a apontar para a posição de memória imediatamente superior. Se em um vetor, o ponteiro passará a apontar a próxima posição do vetor.

`(*p)++`: Incrementa o conteúdo apontado por `p`, ou seja, o valor armazenado na variável para qual `p` está apontando.

`*(p++)`: Incrementa `p` (como em `p++`) e acessa o valor encontrado na nova posição. Se em um vetor, esta expressão acessa o valor da posição imediatamente superior à armazenada em `p` antes do incremento.

2) Solução

`*(p+10)` Acessa o valor encontrado 10 posições a frente de p. Neste caso, o apontador não é incrementado. Se em um vetor, irá acessar a décima posição após a que está sendo apontada.

3) Solução

Dois ponteiros, como outras variáveis, podem ser comparados. Podemos verificar, por exemplo, se dois ponteiros apontam para a mesma posição de memória verificando se `p1 == p2` ou se `p1 != p2`.

Podemos comparar se um ponteiro é “menor” ou “maior” que outro, ou melhor, se aponta para uma posição superior a de outro.

4) Solução

O valor de y é 4, e o programa comentado fica:

```
void main()
{
   int y, *p, x;
   y = 0;      /* atribui o valor 0 a y => y=0 */
   p = &y;     /* atribui o endereço de y ao ponteiro p
                  p contém o endereço de y */
   x = *p;     /* atribui o conteúdo de onde p aponta
                  (valor de y) para x, que passa a valer 0 */
   x = 4;      /* atribui 4 a x */
   (*p)++;     /* incrementa de 1 o conteúdo de onde p aponta,
                  alterando o valor de y para 1 */
   x--;        /* decrementa 1 de x => x = 3 */
   (*p) += x;  /* adiciona x ao conteúdo de onde p aponta,
                  alterando o valor de y para 4 */
   printf("y = %d\n", y);  /* imprime "y = 4" */
}
```

5) Solução

```c
/* Exemplo de função StrLen() */
#include <stdio.h>
int StrLen (char *str)
{
   int tamanho = 0;
   while (*str)
   {
      tamanho++;
      str++;
   }
   return tamanho;
}

int main()
{
   int t;
   char str1[100];
   printf ("\n\nEntre com uma string: \n");
   gets (str1);
   t = StrLen(str1);
   printf ("\n Voce digitou \n%s\ncom %d letras.",str1,t);
   return 0;
}
```

Comentários: A função `StrLen()` percorre o vetor procurando o seu fim. A cada iteração que o fim do vetor não é encontrado, a variável `tamanho` é incrementada. Quando o fim é encontrado, a variável `tamanho` possui o número de iterações usadas até chegar ao fim, ou seja, o tamanho da string.

```c
/* Exemplo de funcao StrCat() */
#include <stdio.h>
#include <string.h>

StrCat (char *primeira, char *segunda)
{
   char *p;
   /* p aponta para o final da primeira string */
   p = primeira + strlen(primeira);
   while (*segunda)
   {
      *p = *segunda;
      p++;
      segunda++;
   }
   *p = '\0';
}
int main ()
{
   char str1[100], str2[1003], str3[200];
   printf("\n\nEntre com uma string: \n");
   gets(str1);
   printf("\n\nEntre com uma string: \n");
   gets(str2);
   str3[0] = '\0';
   StrCat(str3, str1);
   StrCat(str3, str2);
   printf("\nVoce digitou \n%s",str3);
   return(0);
}
```

Comentários: O vetor `p` é apontado para o final da primeira string. Em seguida, cada posição da segunda string é copiada para o endereço `p`, que por sua vez é incrementado a cada iteração. Desta forma, a segunda string é copiada para o final da primeira, e a primeira se torna uma string maior, com o conteúdo das duas strings.

6) Solução

```c
/* Exemplo de funcao strend() */
#include <stdio.h>
#include <string.h>

strend (char *str, char *t)
{
   char *p;
   p = str + strlen(str) - strlen(t); /* p aponta para o final da string str -
                                        tamanho de t */
   while (*t)
   {
      if (*p != *t) return 0;
      p++;
      t++;
   }
   return 1;
}

int main()
{
   char str[100], t[20];
   printf("\n\nEntre com uma string: \n");
   gets(str);

   printf("\n\nEntre com a expressao de teste:\n");
   gets(t);

   if (strend(str, t))
      printf("\n A expressao ocorre no final da frase.\n");
   else
      printf("\n A expressao NAO ocorre no final da frase.\n");
   return(0);
}
```

Comentários: Primeiro, o apontador `p` é apontado para uma posição que será o final da primeira string subtraído do tamanho da string `t`. A partir daí, cada posição é comparada, até o final da string `t`. Se ocorrer alguma posição diferente, a função retorna 0 imediatamente, sem comparar o restante. Caso a função termine o loop, significa que o trecho é realmente igual e a função retorna o flag 1.

7) Solução

O programa contém um erro na linha do `printf()`, onde ele manda imprimir o endereço de `q` (`&q`). Na realidade, para se imprimir o valor `10` (valor de `x`) deve-se imprimir o valor apontado pelo valor apontado por `q`.

Veja passo-a-passo:

`x = 10;`

`p` aponta para `x`;

`q` aponta para `p`;

`*q` é igual a `p`;

como `*p` é igual a `x`, basta escrever `*(*q)` para se ter `x`.

Logo, o `printf()` ficaria: `printf("\n%d\n", **q);`

8) Solução

```c
#include <stdio.h>
#define N 100
```

```
main()
{
   int mat[N][N];
   int *p;
   int i, j, soma = 0;
   p = &mat[0][0];  /* Inicializa o ponteiro no inicio da matriz */

   /* Inicializando a matriz com zeros */
   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
      {
         *p = 0;
         p++;
      }

   /* Preenchendo a matriz com números */
   p = &mat[0][0];
   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
      {
         *p = soma;
         soma++;
         p++;
      }
}
```

Comentários: Pode parecer um pouco inútil este tipo de exercício, mas apesar de não se ter um objetivo claro (uma resposta na tela) ele é útil para desenvolver a capacidade do aluno de trabalhar e resolver problemas usando ponteiros. O exercício aponta o ponteiro `p` para o início da matriz e percorre toda ela preenchendo-a com zeros. Em seguida, `p` é novamente apontado para o início (se não fizer isto, não funciona) da matriz e faz-se então o preenchimento da matriz com os números, usando-se uma variável auxiliar.

9) Solução

(V) Após a atribuição `pa=&a[0];` `pa` e `a` possuem valores idênticos, isto é, apontam para o mesmo endereço.

(V) A atribuição `pa=&a[0];` pode ser escrita como `pa=a;`

(V) `a[i]` pode ser escrito como `*(a+i)`

(V) `&a[i]` e `a+i` são idênticos

(V) `a+i` é o endereço do i-ésimo elemento do vetor `a`

(V) `pa[i]` é idêntico a `*(pa+i)`

(V) `pa=a` é uma operação válida

(V) `pa++` é uma operação válida

(F) `a=pa` é uma operação válida ==> Um vetor não pode ter seu endereço modificado

(F) `a++` é uma operação válida ==> Um vetor não pode ter seu endereço modificado

Comentários

- Se `pa` aponta para um elemento particular de um vetor, então, por definição, `pa+1` aponta para o próximo elemento, `pa+i` aponta para `i` elementos após `pa` e `pa-i` aponta para `i` elementos antes de `pa`. Assim, se `pa` aponta para `a[0]`, `*(pa+1)` refere-se ao conteúdo de `a[1]`, `pa+i` é o endereço de `a[i]` e `*(pa+i)` é o conteúdo de `a[i]`.
- Estas observações aplicam-se independentemente do tipo ou tamanho das variáveis no vetor `a`.
- A correspondência entre indexação e aritmética com ponteiros é evidentemente muito estreita. Por definição, o valor de uma variável ou expressão do tipo vetor é o endereço do elemento zero do vetor.

Assim, após a atribuição `pa=&a[0];` `pa` e `a` possuem valores idênticos. Como o nome de um vetor é sinônimo para o local do elemento inicial, a atribuição `pa=&a[0]` também pode ser escrita como `pa=a;`

- `a[i]` pode ser escrita como `*(a+i)`. Na avaliação de `a[i]`, o C a converte para `*(a+i)` imediatamente; as duas formas são equivalentes. Aplicando-se o operador `&` a ambas as partes dessa equivalência, segue-se que `&a[i]` e `a+i` também são idênticos: `a+i` é o endereço do i-ésimo elemento após `a`. Por outro lado, se `pa` é um apontador, expressões podem usá-lo como um subscrito; `pa[i]` é idêntico a `*(pa+i)`. Em suma, qualquer expressão de vetor e índice é equivalente a uma escrita com um apontador e um deslocamento.
- Existe uma diferença importante entre o nome de um vetor e um apontador que deve ser sempre lembrada: um apontador é uma variável, de forma que `pa=a` e `pa++` são operações válidas. Mas o nome de um vetor não é uma variável (poderia ser visto como um ponteiro constante, mas não uma variável); construções como `a=pa` e `a++` são ilegais.

10) Solução

a) Programa 1

Este programa atribui o valor 10 a alguma posição de memória desconhecida. O ponteiro `p` nunca recebeu um valor; portanto, ele contém lixo. Esse tipo de problema sempre passa despercebido, quando o programa é pequeno, porque as probabilidades estão a favor de que `p` contenha um endereço "seguro" – um endereço que não esteja em seu código, área de dados ou sistema operacional. Contudo, a medida que seu programa cresce, a probabilidade de `p` apontar para algo vital aumenta. Eventualmente, seu programa para de funcionar. A solução é sempre ter certeza de que um ponteiro está apontando para algo válido antes de usá-lo.

b) Programa 2

O erro aqui apresentado é provocado por um simples equívoco sobre como usar um ponteiro. A chamada de `printf()` não imprime o valor de `x`, que é 10. Imprime um valor desconhecido porque a atribuição `p=x;` está errada. Esse comando atribui o valor 10 ao ponteiro `p`, que se supõe conter um endereço, não um valor. Para corrigi-lo, basta escrever `p=&x;`

QUESTÕES PROPOSTAS

1) Escrever um programa para imprimir as matrizes declaradas abaixo utilizando ponteiros.

```
# include <stdio.h>
main()
{
       int vetor [5] = { 0, 1, 2, 3, 4};
       int matriz [5][5] = { 0, 0, 0, 0,
                             1, 1, 1, 1,
                             2, 2, 2, 2,
                             3, 3, 3, 3,
                             4, 4, 4, 4 };
      char string [10] = "BICAMPEAO";
}
```

2) Qual será a saída deste programa supondo que i ocupa o endereço 2000 na memória?

```c
#include <stdio.h>

int main(void)
{
   int i=5, *p;
   p = &i;
   printf("%u %d %d %d \n", p, *p+2, **&p, 3**p, **&p+4);
   return 0;
}
```

3) Seja o seguinte trecho de programa:

```c
int i=3, j=5;
int *p, *q;
p = &i;
q = &j;
```

Qual é o valor das expressões?

a. p==&i

b. *p-*q

c. **&p

d. 3*-*p/(*q)+7

4) Assumindo que pulo[] é uma matriz de uma dimensão do tipo int, quais das seguintes expressões referenciam o valor do terceiro elemento da matriz?

a. *(pulo + 3)

b. *(pulo + 2)

c. *(pulo + 4)

d. pulo + 3

e. pulo + 2

5) O que faz o programa seguinte quando executado?

```c
#include <stdio.h>
int main(void)
{
   static int mat[] = {4, 5, 6};
   int i;
   for (i=0; i<3; i++)
      printf("%d ", *(mat + i));
   return 0;
}
```

6) O que faz o programa seguinte quando executado?

```c
#include <stdio.h>
int main(void)
{
   static int mat[] = {4, 5, 6};
   int i, *ptr;
   ptr = mat;
   for (i=0; i<3; i++)
      printf("%d ", *ptr++);
   return 0;
}
```

7) Considere as declarações:

```
int vetor[10];
int *ponteiro;
```

Diga quais expressões abaixo são válidas ou não (neste caso, explique):

a. vetor = vetor + 2;

b. vetor++;

c. vetor = ponteiro;

d. ponteiro = vetor;

e. ponteiro = vetor+2;

8) Escreva uma função que receba um ponteiro para uma cadeia de caracteres e troque todo o caractere após um espaço em branco pelo seu equivalente maiúsculo.

9) Declare vetores de inteiro, char, float, double, long int com 5 posições, da seguinte forma:

```
int vi[5] = { 2, 5, 1, 4, 0};
char vc[5] = { 'a', 'b', 'm', '4', '-'};
float vf[5] = {2.66, 0.125, 1.0, 4.99, 2.009};
double vd[5] = { 1.33, 2.66, 3.99, 4.33, 5.66};
```

Usando a função `printf()`, com o argumento "`%p`" (Repare que o valor impresso é um inteiro, porém expresso na base hexadecimal), e com vetores apontando para cada um dos tipos de dados, descubra quantos bytes é alocado pelo seu compilador a cada tipo de dados. Não se esqueça que para um vetor apontar para um vetor de `float`, ele deve ser do tipo `float`. Como se sabe que em um vetor as posições são contínuas, se for impresso o endereço de duas posições, pela diferença entre os dois endereços pode-se descobrir quantos bytes são alocados.

10) Declare um vetor de inteiros com 300 posições, de forma que cada posição possua o valor igual o índice da posição (logo, o vetor será ordenado de 0 a 299). Declare um ponteiro que aponte para a quinta posição deste vetor, ou seja,

```
p = &v[5];
```

Usado a função `printf()`, e o endereçamento do tipo `*(p ± n)`, imprima o valor de todos os elementos do vetor, com respectivos endereços. Não confundir com o endereçamento `*p+n`, que imprime o conteúdo de `p` somado a `n`.

11) Explique o que faz o seguinte trecho de código

```
float f = 10.5;
int i = *(int*)&f;
```

Cap. 9: FUNÇÕES: ASPECTOS AVANÇADOS

EXERCÍCIOS DE FIXAÇÃO

1) Qual a afirmativa verdadeira?

a. Você pode retornar para um programa quantas variáveis de uma função desejar através do comando `return`.

b. Uma função só pode ter um comando `return`.

c. Os protótipos de função servem para declarar as funções, isto é, indicar para o compilador qual o seu nome, tipo de retorno e o número e tipos dos parâmetros.

d. Uma função não pode retornar um ponteiro.

e. Nenhuma das opções anteriores.

2) Qual das seguintes razões não é uma razão válida para o uso de funções em C?

a. Funções usam menos memória do que se repetirmos o mesmo código várias vezes.

b. Funções rodam mais rápido.

c. Funções fornecem um meio de esconder cálculos em uma "caixa preta" que pode ser usada sem a preocupação de detalhes internos de implementação.

d. Funções mantêm variáveis protegidas das outras partes do programa.

3) Qual a afirmativa falsa?

a. Se uma função não retorna nada ela deve ser declarada como `void`.

b. O retorno da função main é feito para o sistema operacional.

c. stdio.h e string.h contêm o protótipo de algumas funções da biblioteca do C.

d. Funções podem ser definidas dentro de outras funções.

e. Uma das opções anteriores é falsa.

4) Qual a afirmativa verdadeira?

a. `stdio.h` e `string.h` contêm o corpo de algumas funções da biblioteca do C.

b. Funções podem ser chamadas por outras funções.

c. Em um programa C todas as funções de um programa devem estar em um único arquivo .c.

d. Variáveis declaradas em uma função são acessíveis por todas as outras funções.

e. Nenhuma das opções anteriores.

5) Qual a afirmativa verdadeira?

a. A palavra reservada `auto` é utilizada para dizer que uma variável é local (automática). Porém, ela pode ser omitida dentro de uma função, pois todas as variáveis são locais por default.

b. Não se pode utilizar variáveis com o mesmo nome em funções diferentes.

c. Os parâmetros recebidos por uma função têm o mesmo endereço das variáveis usadas na chamada à função.

d. Quando uma variável local tem o mesmo nome de uma variável global, a variável local se torna inacessível e a variável global é acessível.

e. Nenhuma das opções anteriores.

6) Qual a afirmativa falsa?

a. Os parâmetros recebidos por uma função armazenam cópias das variáveis usadas na chamada da função.

b. Variáveis globais são conhecidas e podem ser alteradas por todas as funções do programa.

c. Quando queremos alterar as variáveis que são passadas como parâmetros para uma função, devemos declará-las como ponteiros na função.

d. A função `scanf()` necessita receber como parâmetro o endereço da variável de entrada, porque ela precisa alterar esta variável.

e. Uma das opções anteriores é falsa.

7) O que imprime o programa abaixo?

```
#include <stdio.h>

void func();

int i = 10;

void main()
{
   int i=20;
   func();
   printf("i= %d ", i);
   {
      int i = 30;
      func();
      printf("i= %d ", i);
   }
}

void func()
{
   printf("i = %d ", i);
}
```

a. i= 20 i= 20 i= 30 i= 30

b. i= 10 i= 20 i=10 i= 30

c. i= 20 i=10 i=10 i=30

d. i= 10 i=10 i=10 i=10

e. Nenhuma das opções anteriores.

8) Ao se utilizar um vetor como parâmetro para uma função que informação está sendo passada à função?

a. Uma cópia de todos elementos do vetor.

b. Uma cópia do primeiro elemento do vetor.

c. O endereço do primeiro elemento do vetor.

d. O endereço de todos os elementos do vetor.

e. Nenhuma das opções anteriores.

9) Sejam par1, par2 e par3 variáveis inteiras. Se chamarmos uma função pela instrução:

```
func(&par1, &par2, &par3);
```

Para que servem `&par1`, `&par2` e `&par3`?

a. São valores inteiros passados para a função.

b. Servem para armazenar os endereços da função e das funções que chamaram.

c. São os endereços das variáveis da função que chamou. Nestes endereços iremos armazenar os valores a serem modificados pela função.

d. Armazenam os endereços das funções de biblioteca usados na função.

e. Nenhuma das opções anteriores.

10) O que imprime o programa a seguir?

```c
#include <stdio.h>

func(int *a, int b)
{
   int temp;
   temp = *a;
   *a = b;
   b = temp;
}

void main()
{
   int a = 10, b = 20;
   func(&a, b);
   printf("a = %d, b = %d", a, b);
}
```

a. a = 10, b = 20

b. a = 20, b = 10

c. a = 10, b = 10

d. a = 20, b = 20

e. Nenhuma das opções anteriores.

Gabarito: 1-c; 2-b; 3-d; 4-b; 5-a; 6-e; 7-b; 8-c; 9-c; 10-d.

Exercícios Resolvidos

Enunciados

1) Faça o que se pede:

a. Construa uma função `encaixa` que dados dois inteiros positivos `a` e `b` verifica se `b` corresponde aos últimos dígitos de `a`.

Exemplo:

a	b	
567890	890	=> encaixa
1243	1243	=> encaixa
2457	245	=> não encaixa
457	2457	=> não encaixa

b. Usando a função do item anterior, faça um programa que lê dois inteiros positivos `a` e `b` e verifica se o menor deles é segmento do outro.

Exemplo:

a	b	
567890	678	=> b é segmento de a
1243	2212435	=> a é segmento de b
235	236	=> um não é segmento do outro

2) Resolva o que se pede:

a. Faça uma função `arctan` que recebe o número real `x` ∈ [0,1] e devolve uma aproximação do arco tangente de `x` (em radianos) através da série

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

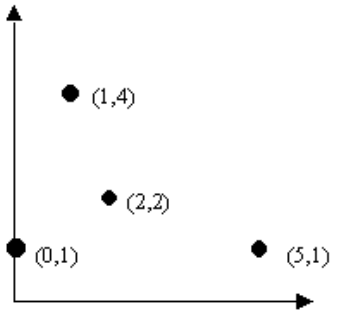
incluindo todos os termos da série até

$$\left|\frac{x^k}{k}\right| < 0{,}0001$$

b. Faça uma função `angulo` que recebe um ponto de coordenadas cartesianas reais (*x*,*y*), com $x \geq 0$ e $y \geq 0$ e devolve o ângulo formado pelo vetor (*x*,*y*) e o eixo horizontal.

Exemplos: Observe a figura abaixo e verifique que os ângulos correspondentes aos pontos marcados é de aproximadamente

(0,1)	90 graus
(2,2)	45 graus
(1,4)	75 graus
(5,1)	11 graus



Use a função do item anterior. Note que a função só calcula o arco tangente de números entre 0 e 1, e o valor devolvido é o ângulo em radianos (use o valor π = 3.14 radianos = 180 graus).

Para calcular o valor do ângulo α pedido, use a seguinte fórmula:

$$\alpha = \begin{cases} arctan\left(\frac{y}{x}\right) \; caso \; y < x \\ \frac{\pi}{2} - arctan\left(\frac{x}{y}\right) \; caso \; contrário \end{cases}$$

c. Faça um programa que, dados *n* pontos do primeiro quadrante ($x \geq 0$ e $y \geq 0$) através de suas coordenadas cartesianas, determina o ponto que forma o menor ângulo com o eixo horizontal. Use a função do item anterior.

3) Escreva um programa que leia um vetor de inteiros pelo teclado e o apresente na tela. Crie uma função `void LeVetor(int *vet, int dimensao)` para fazer a leitura do vetor.

4) Escreva uma função que receba duas strings como argumentos e troque o conteúdo da primeira string com o conteúdo da segunda.

5) Faça um programa que utilize os argumentos `argv` e `argc`. O programa deverá receber dois números e apresentar a soma dos dois. Veja que para isto você deverá ter também uma função que transforme uma string em um inteiro, pois o tipo de `argv` é `char`; logo você irá receber strings e deverá transformá-las em inteiros antes de somá-las.

6) Um problema tradicional é o de encontrar o enésimo termo da série de Fibonacci. As séries de Fibonacci são de grande importância matemática, e a lei básica é que a partir do terceiro termo, todos os termos são a soma dos dois últimos. Os primeiros termos da sequência são: 1, 1, 2, 3, 5, 8, 13, 21, 34... O primeiro e o segundo termos são 1. O terceiro termo é 2 (1+1). O quarto termo é 3 (1+2). O quinto termo é 5 (2+3) ... Faça uma função que encontre o enésimo termo da sequência de Fibonacci. **Use recursividade**.

Respostas e Comentários

1) Solução

```c
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int encaixa(int a, int b)
{

   while (b != 0 && a % 10 == b % 10)
   {
      a = a/10;
      b = b/10;
   }

   if (b == 0)
      return TRUE;
   else
      return FALSE;

}

main()
{
   int a, b, subseq, maior, menor;

   printf("Digite dois inteiros positivos: ");
   scanf("%d %d", &a, &b);

   if (a > b)
   {
      maior = a;
      menor = b;
   }
   else
   {
      maior = b;
      menor = a;
   }

   subseq = FALSE;
   while (maior >= menor)
   {
      if (encaixa(maior,menor) == TRUE) subseq = TRUE;
      maior = maior/10;
   }
   if (subseq == TRUE)
      printf("%d eh segmento de %d\n", b, a);
   else
      printf("%d nao eh segmento de %d\n", b, a);
}
```

2) Solução

```c
#include <stdio.h>

#define PI 3.14

/* Item (a) */
float arctan(float x)
{
   int k;
   float termo, soma, eps;

   eps = 0.0001;
   termo = x;
   soma = x;
   k = 1;

   while (termo*termo > eps*eps)
   {
      k = k+2;
      termo = -termo*x*x*(k-2)/k;
      soma = soma + termo;
   }

   return soma;
}


/* Item (b) */
/* Funcao extra: transforma angulo de radianos para graus */

float graus(float x)
{
   return 180*x/PI;
}


float angulo(float x, float y)
{
   float resp;

   if (x == 0 && y == 0) /* ERRO??? */
      return -1;

   if (x >= y)
      if (x != 0)
         resp =  graus(arctan(y/x));
      else
         resp = 90;
   else
      if (y != 0)
         resp = graus(PI/2 - arctan(x/y));
      else
         resp = 0;

   return resp;
}


int main()
{
   int i, n;
   float x, y, xmin, ymin, angmin, ang;

   printf("Digite quantidade de pontos: ");
   scanf("%d", &n);

   angmin = 900;
   xmin = 0;
   ymin = 0;
```

```
   for (i = 0; i < n; i++)
   {
      printf("Digite as coordenados de um ponto: ");
      scanf("%f %f", &x, &y);
      ang = angulo(x,y);
      if (ang == -1)
         printf("Ponto na origem, sera desconsiderado\n");
      else
         if (ang < angmin)
         {
            angmin = ang;
            xmin = x;
            ymin = y;
         }
   }

   printf("O ponto de menor angulo: (%f, %f)\n", xmin, ymin);
   printf("Menor angulo: %f graus\n", angmin);

   return 0;
}
```

3) Solução

```
#include <stdio.h>
#define MAX 10 /* dimensão do vetor */

void LeVetor(int *vet, int dimensao); /* protótipo de LeVetor */

int main()
{
   int i;
   int vet[MAX];
   LeVetor(vet, MAX); /* Para passar um vetor como parametro é necessario
                         apenas passar o endereco de seu primeiro elemento */
   printf("Os numeros lidos foram \n");
   for (i=0; i<MAX; i++) printf("%d ", vet[i]);

   return 0;
}

/* Endereco de inicio do vetor, tamanho do vetor */
void LeVetor(int *vet, int dimensao )
{
   int i;
   printf("Digite os %d valores a serem armazenados no vetor ", dimensao);
   for (i=0; i < dimensao; i++) scanf("%d", (vet+i));
}
```

4) Solução

```
#include <stdio.h>
#include <string.h>
#define STRMAX 50

void TrocaString(char *str_a, char *str_b)
{
   char strtemp[STRMAX]; /* string temporaria usada para a troca */
   strcpy (strtemp, str_a);
   strcpy (str_a, str_b);
   strcpy (str_b, strtemp);
}

int main ()
{
   char string1[STRMAX], string2[STRMAX];
   printf ("Entre com a primeira string: ");
   gets (string1);
```

```
   printf ("\nEntre com a segunda string: ");
   gets (string2);
   TrocaString(string1, string2);
   printf ("\nA primeira string agora eh: %s\nA segunda string agora eh: %s",
      string1, string2);
   return 0;
}
```

5) Solução

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
   int numa, numb;
   if (argc !=3)
      printf("Para usar, voce tem que passar dois parametros \n");
   else
   {
      if (((numa = atoi(argv[1])) == 0) || ((numb = atoi(argv[2])) == 0))
         printf("\n\nArgumentos invalidos!! \n");
      else
         printf("\n\nA soma de %d e %d eh %d\n\n",numa, numb, numa+numb);

   }
   return 0;
}
```

Comentários: Para a transformação dos números que estavam em `argv[]` para `int`, usou-se a função `atoi()` da biblioteca `stdlib.h`, que recebe um argumento do tipo `char*` e retorna o `int` correspondente, caso a string represente um número. Caso o valor retornado por `atoi` seja nulo, indica que o argumento passado foi inválido.

6) Solução

```
#include <stdio.h>

int fib(int n)
{
   if (n>2) return fib(n-1)+fib(n-2);
   else return 1;
}

int main()
{
   int n, fibon;
   printf("\n\nEntre com um numero: ");
   scanf("%d", &n);
   fibon = fib(n);
   printf("\nO termo %d da serie de Fibonacci eh: %d\n", n, fibon);
   return 0;
}
```

QUESTÕES PROPOSTAS

1) Escreva uma função `desenhaQuadrado` que exibe um quadrado sólido (o mesmo número de linhas e colunas). O caractere utilizando para preencher o quadrado e o valor do lado são passados como argumentos para a função. Por exemplo, se o caractere for `x` e o valor do lado for 5, a função deverá exibir:

```
xxxxx
xxxxx
xxxxx
xxxxx
xxxxx
```

2) Elabore uma função que receba uma matriz N x M de inteiros como parâmetro e imprima na tela as seguintes informações, a serem implementadas de maneira modularizada (usando outras funções):
   a. Média dos elementos de cada linha;
   b. Maior e Menor elemento de cada linha;

3) Crie uma função `float round(float r)` que faça o arredondamento de números reais. Por exemplo: 5 = round (5.4), 7 = round(6.5).

4) Escreva um programa que leia um vetor gabarito de 10 elementos. Cada elemento de gabarito contem um caractere de `'a'` a `'e'` correspondente as opções corretas de uma prova objetiva. Em seguida o programa deve ler um vetor resposta, também de 10 elementos do tipo caractere, contendo as respostas de um aluno. O programa deve comparar os dois vetores (através de uma função que retorna o número de acertos) e escrever a nota do aluno.

5) Implementar a função `unsigned int inverte (unsigned int n)` que recebe um número `n` como parâmetro e retorna este número escrito ao contrário. Ex: 431 <-> 134. Sugestão: Encontre a ordem de grandeza do número primeiro, usando um loop. Depois, vá efetuando sucessivas divisões inteiras a fim de encontrar os valores absolutos que compõem o número, ao mesmo tempo que você vai efetuando a composição inversa do número.

6) Faça uma função que recebe, por parâmetro, a hora de inicio e a hora de término de um jogo, ambas subdivididas em 2 valores distintos: horas e minutos. A função deve retornar a duração do jogo em minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode começar em um dia e terminar no outro.

7) Escreva uma função que recebe duas variáveis inteiras “zera” seus valores.

8) Crie uma função que receba 3 vetores de inteiros com N elementos (use a diretiva `#define`) como parâmetros e “preencha” o terceiro vetor com a soma dos outros dois (elemento a elemento). Crie uma função `main` que faça uso apropriado desta função.

9) Crie uma função que receba 3 matrizes de inteiros com N x M elementos (use a diretiva `#define`) como parâmetros e “preencha” a terceira matriz com a soma das outras duas (elemento a elemento). Crie uma função `main` que faça uso apropriado desta função.

10) Escreva uma função que receba uma matriz NxN como parâmetro e retorne o a média dos elementos da sua diagonal principal.

11) Escreva um programa que solicita a usuário que escreva seu primeiro nome seguido de um espaço em branco e após a sua data de nascimento, no formato dd/mm/aaaa. Capture tudo isso numa única string. Em seguida, separe o nome, dia, mês e ano de nascimento (`sscanf`) e “imprima” numa outra string (`sprintf`) de acordo com o exemplo abaixo. Exiba a string final construída.

    ```
    Jose 25/10/1994
    Ola Jose, voce nasceu no dia 25 de outubro de 1994.
    ```

12) Elabore os seguintes programas usando **recursão**:
    a. Dado um número natural `n`, exiba uma contagem regressiva de `n` até `0`.
    b. Dado um número natural `n`, exiba uma contagem progressiva de `1` até `n`.
    c. Calcule o somatório de `1` a um número inteiro `n`.
    d. Calcule `x` elevado a `n`, onde `n` é sempre inteiro e maior que `0`.
    e. Dados dois números naturais `m` e `n`, calcule o produto de `m` por `n` usando apenas soma e subtração.
    f. Dados dois números naturais `m` e `n` $\neq$ `0`, calcule o resto da divisão inteira de `m` por `n` usando apenas subtração.
    g. Dado um número natural `n`, determine se `n` é par usando apenas subtração.

Cap. 10: Tipos de Dados Avançados e Definidos Pelo Usuário

Questões Propostas

1) Escreva um programa para armazenar num array um certo número de inteiros. O programa deverá começar por perguntar ao utilizador o número de inteiros a armazenar; deverá de seguida alocar dinamicamente um array de inteiros capaz de armazenar esse número de inteiros; e por fim deverá lê-los do teclado e reescrevê-los no vídeo. Antes de terminar não se esquecer de libertar a memória alocada.

2) Escreva um programa que leia dois vetores de mesmo tamanho. No `main`, o usuário deverá informar o tamanho e os elementos dos vetores. Escrever uma função que receba como parâmetros os vetores lidos e gere um novo vetor que corresponde aos vetores passados como parâmetro de forma intercalada. Retorne este vetor gerado para a função `main`. Exiba este vetor na tela.

3) Escreva um programa que leia um vetor com N elementos inteiros a partir da função `main`. A seguir, em uma função receba este vetor e separe-o em dois outros vetores, um com os elementos positivos e zero e outro com os valores negativos. Esses dois últimos vetores devem ter seus tamanhos ajustados para o número exato de elementos a serem armazenados. "Retorne", via parâmetros recebidos por referência, ambos os vetores para o `main`. Exiba ambos os vetores.

4) Implemente um programa que lê o nome e a pontuação de M candidatos de um concurso e ao final exiba a listagem (nome e pontuação) dos candidatos com as N maiores pontuações. M e N devem ser fornecidos pelo usuário. Deve ser criada uma estrutura com os campos nome (`string`) e score (`float` positivo) para armazenar os dados de um candidato. Também deve ser definida uma função que recebe um vetor de candidatos, que foram capturados na função `main`, e retorna um vetor com os candidatos com as N maiores pontuações. (Adapte alguma função de ordenação para ordenar esse vetor de structs)

Exercícios Resolvidos

Enunciados

1) Faca um programa que multiplique duas matrizes. O programa deverá estar estruturado de maneira que:
   - O usuário forneça as dimensões das matrizes (teste se as dimensões são compatíveis, isto é, se as matrizes podem ser multiplicadas);
   - As matrizes sejam alocadas dinamicamente (de acordo com o Cap. 10 da apostila);
   - As matrizes sejam lidas pelo teclado (faça uma função para leitura das matrizes);
   - As matrizes sejam, então, multiplicadas (faça uma função para a multiplicação);
   - A matriz resultante seja apresentada em tela (faça uma função para apresentar a matriz na tela).

   OBS:

   a) Faça, também, a alocação dinâmica da matriz resultante.
   b) Caso alguém não conheça o procedimento para a multiplicação de matrizes, segue aqui a orientação.

Suponha as matrizes

$$A_{m \times n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

$$B_{n \times t} = \begin{bmatrix} b_{11} & \cdots & b_{1t} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nt} \end{bmatrix}$$

O elemento `ij` da matriz C é resultante da multiplicação da linha `i` de A pela coluna `j` de B. Portanto, a matriz $C_{m \times t} = A * B$ será da seguinte forma:

$$C_{m \times t} = \begin{vmatrix} (a_{11}*b_{11}+a_{12}*b_{21}+...+a_{1n}*b_{n1}) & (a_{11}*b_{12}+a_{12}*b_{22}+...+a_{1n}*b_{n2}) & ... & (a_{11}+b_{1t}+a_{12}*b_{2t}+...+a_{1n}*b_{nt}) \\ (a_{21}*b_{11}+a_{22}*b_{21}+...+a_{2n}*b_{n1}) & (a_{21}*b_{12}+a_{22}*b_{22}+...+a_{2n}*b_{n2}) & ... & (a_{21}+b_{1t}+a_{22}*b_{2t}+...+a_{2n}*b_{nt}) \\ ... & ... & ... & \\ (a_{m1}*b_{11}+a_{m2}*b_{21}+...+a_{mn}*b_{n1}) & (a_{m1}*b_{12}+a_{m2}*b_{22}+...+a_{mn}*b_{n2}) & ... & (a_{m1}+b_{1t}+a_{m2}*b_{2t}+...+a_{mn}*b_{nt}) \end{vmatrix}$$

2) Escreva um programa fazendo uso de `struct`'s. Você deverá criar um `struct` chamado `Ponto`, contendo apenas a posição `x` e `y` (inteiros) do ponto. Declare 2 pontos, leia a posição (coordenadas `x` e `y`) de cada um e calcule a distância entre eles. Apresente no final a distância entre os dois pontos.

3) Seja o seguinte `struct` que é utilizado para descrever os produtos que estão no estoque de uma loja:

```
struct Produto {
    char nome[30]; /* Nome do produto */
    int codigo;    /* Codigo do produto */
    double preco;  /* Preco do produto */
};
```

   a) Escreva uma instrução que declare uma matriz de Produto com 10 itens de produtos;

   b) Atribua os valores "Pe de Moleque", 13205 e R$0,20 aos membros da posição 0 e os valores "Cocada Baiana", 15202 e R$0,50 aos membros da posição 1 da matriz anterior;

   c) Faça as mudanças que forem necessárias para usar um ponteiro para Produto ao invés de uma matriz de Produtos. Faça a alocação de memória de forma que se possa armazenar 10 produtos na área de memória apontada por este ponteiro e refaça as atribuições da letra b;

   d) Escreva as instruções para imprimir os campos que foram atribuídos na letra c.

4) Crie uma estrutura chamada retângulo, que possua duas estruturas ponto (o ponto superior esquerdo e o ponto inferior direito). Faça um programa que receba (via teclado ou arquivo) as informações acerca de um retângulo (as coordenadas dos dois pontos), e informe dados interessantes sobre o retângulo, como a área, o comprimento da diagonal e o comprimento de cada aresta.

5) Faça um exercício usando enumeração. Crie uma enumeração de meses do ano, e a use para indexar um vetor de nomes dos meses. Desta forma, apresente os nomes dos meses do ano na tela.

RESPOSTAS E COMENTÁRIOS

1) Solução

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

float **Alocar_matriz_real(int m, int n)
{
   float **v; /* ponteiro para a matriz */
   int i; /* variavel auxiliar */
   if (m < 1 || n < 1) { /* verifica parametros recebidos */
      printf("** Erro: Parametro invalido **\n");
      return (NULL);
   }

   /* aloca as linhas da matriz */
   v = (float **) calloc (m, sizeof(float *));
   if (v == NULL) {
      printf ("** Erro: Memoria Insuficiente **");
      return (NULL);
   }

   /* aloca as colunas da matriz */
   for ( i = 0; i < m; i++ ) {
      v[i] = (float*) calloc (n, sizeof(float));
      if (v[i] == NULL) {
         printf ("** Erro: Memoria Insuficiente **");
         return (NULL);
      }
   }
   return (v); /* retorna o ponteiro para a matriz */
}

float **Liberar_matriz_real(int m, int n, float **v)
{
   int i; /* variavel auxiliar */
   if (v == NULL) return (NULL);
   if (m < 1 || n < 1) { /* verifica parametros recebidos */
      printf ("** Erro: Parametro invalido **\n");
      return (v);
   }
   for (i=0; i<m; i++) free (v[i]); /* libera as linhas da matriz */
   free (v); /* libera a matriz */
   return (NULL); /* retorna um ponteiro nulo */
}

void Le_matriz_real(int linhas, int colunas, float **matriz)
{
   int i, j;
   for (i = 0; i < linhas; i++)
   {
      printf("\nlinha %d: \n", i+1);
      for (j= 0; j<colunas; j++)
         scanf("%f", &matriz[i][j]);
   }
}

void Multiplica_matriz_real(int linha3, int coluna3, int linha2,
                            float **mat1, float **mat2, float **mat3)
{
   int i, j, t;
   for(i=0; i< linha3; i++)
      for(j=0; j< coluna3; j++)
      {
         mat3[i][j] = 0;
         for(t=0; t< linha2; t++) /* linha2, que e igual a coluna1.. */
            mat3[i][j] += mat1[i][t]*mat2[t][j];
      }
}
```

```
void Imprime_matriz_real(int linha, int coluna, float **mat)
{
   int i,j;
   for (i =0; i < linha; i++)
   {
      for (j=0; j<coluna; j++)
         printf("%f\t", mat[i][j]);
      printf("\n");
   }
}

void main (void)
{
   float **mat1, **mat2, **mat3; /* matrizes a serem alocadas */
   int linha1, coluna1; /* Dimensoes das matrizes */
   int linha2, coluna2;
   int linha3, coluna3;
   int erro=0;
   printf("\n\n-------- Multiplicacao de Matrizes: ----------\n");
   printf("Alocacao Dinamica de Memoria\n");
   printf("-----------------------------------------------\n");

   /* Le e compara as dimensoes das matrizes.
   So abandona o loop se as dimensoes forem validas
   Atencao aa condicao... */
   do
   {
      printf("\nDimensoes da matriz 1 (linhas e colunas): ");
      scanf("%d%d", &linha1, &coluna1);
      printf("\nDimensoes da matriz 2 (linhas e colunas): ");
      scanf("%d%d", &linha2, &coluna2);
      if ( coluna1 != linha2 )
         printf("\nDimensoes Invalidas! Tente de novo..\n");
   } while ( coluna1 != linha2 );

   /* Dimensoes da matriz de resposta: */
   linha3 = linha1;
   coluna3 = coluna2;

   /* Aloca a memoria para as matrizes: */
   if ((mat1 = Alocar_matriz_real (linha1, coluna1))== NULL) erro = 1;
   if ((mat2 = Alocar_matriz_real (linha2, coluna2))== NULL) erro = 1;
   if ((mat3 = Alocar_matriz_real (linha3, coluna3))== NULL) erro = 1;
   if (erro)
   {
      printf("\n Memoria Insuficiente! Abortando..\n");
      exit(1);
   }
   /* Le a Matriz 1: */
   printf("\n\nDigite a Matriz 1:\n");
   Le_matriz_real(linha1,coluna1,mat1);
   /* Le a Matriz 2: */
   printf("\n\nDigite a Matriz 2:\n");
   Le_matriz_real(linha2,coluna2,mat2);
   /* Imprime as matrizes lidas */
   printf ("\n\n==>Matriz 1\n");
   Imprime_matriz_real(linha1,coluna1,mat1);
   printf ("\n\n==>Matriz 2\n");
   Imprime_matriz_real(linha2,coluna2,mat2);

   /* --------------------- Multiplicacao..*/
   Multiplica_matriz_real(linha3,coluna3,linha2,mat1,mat2,mat3);
   /* ------------- Imprime a Matriz Calculada... */
   printf("\n\n==> Matriz 3 , Resultado da multiplicacao:\n");
   Imprime_matriz_real(linha3, coluna3, mat3);

   Liberar_matriz_real(linha1, coluna1, mat1);
   Liberar_matriz_real(linha2, coluna2, mat2);
   Liberar_matriz_real(linha3, coluna3, mat3);
}
```

2) Solução

```c
#include <stdio.h>
#include <math.h> /* Para as funções de raiz e potência */

struct ponto {
   int x;
   int y;
};

void le_ponto(struct ponto *p, char *);
float dist(struct ponto p1, struct ponto p2);

void main(void)
{
   struct ponto p1, p2;
   printf("\nDistancia entre os pontos:\n");
   le_ponto(&p1, "primeiro");
   le_ponto(&p2, "segundo");
   printf("\n\nDistancia entre os pontos: %5.2f\n", dist(p1, p2));
}

void le_ponto(struct ponto *p, char *s)
{
   int x, y;
   printf("Digite as coordenadas do %s ponto (x,y): ", s);
   scanf("%d%d", &x, &y);
   p->x = x;
   p->y = y;
}

float dist(struct ponto p1, struct ponto p2)
{
   float s1, s2;
   s1 = pow((p1.x-p2.x), 2); /* Função pow(x,y) retorna x^y */
   s2 = pow((p1.y-p2.y), 2);
   return sqrt( s1 + s2);    /* Função sqrt(x) retorna a raiz quadrada de x */
}
```

3) Solução

a) A instrução abaixo faz o que foi pedido

```c
struct Produto prod[10]; /* Declara prod como um vetor que armazena 10 produtos */
```

b) O programa a seguir faz o que foi pedido:

```c
#include <string.h>

struct Produto {
   char nome[30]; /* Nome do produto */
   int codigo;    /* Codigo do produto */
   double preco;  /* Preco do produto */
};

int main()
{
   struct Produto prod[10];
   strcpy(prod[0].nome,"Pe de Moleque");
   prod[0].codigo = 13205;
   prod[0].preco = 0.20;
   strcpy(prod[1].nome,"Cocada Baiana");
   prod[1].codigo = 15202;
   prod[1].preco = 0.50;
   return(0);
}
```

c) e d) O programa a seguir faz o que foi pedido, incluindo a impressão:

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

struct Produto {
   char nome[30]; /* Nome do produto */
   int codigo;    /* Codigo do produto */
   double preco;  /* Preco do produto */
};

int main()
{
   struct Produto *prod;
   int i;
   prod = (struct Produto *) malloc(10*sizeof(struct Produto));
   if (prod == NULL)
   {
      printf("\n Memoria Insuficiente");
      exit(1);
   }
   strcpy(prod[0].nome,"Pe de Moleque");
   prod[0].codigo = 13205;
   prod[0].preco = 0.20;
   strcpy(prod[1].nome,"Cocada Baiana");
   prod[1].codigo = 15202;
   prod[1].preco = 0.50;
   printf("Quitanda do Manuel: Lista de Produtos");
   for(i=0; i <= 1; i++)
   {
      printf("\n\nProd %d: %s",i+1, prod[i].nome);
      printf("\nCodigo: %d",prod[i].codigo);
      printf("\nPreco : R$ %.2f", prod[i].preco);
   }
   return 0;
}
```

4) Solução

```
#include <stdio.h>
#include <math.h>
typedef struct _ponto /* Aqui usa-se typedef para dar o nome desejado ao tipo */
{
   int x;
   int y;
} Ponto;

typedef struct _retangulo
{
   Ponto sup_esq;
   Ponto inf_dir;
} Retangulo;

void le_ponto(Ponto *p, char *);
float diagonal(Retangulo r);
int area(Retangulo r);
void arestas(Retangulo r);

void main(void)
{
   Retangulo r;
   printf("\nPontos do retangulo:\n");
   le_ponto(&(r.sup_esq), "primeiro");
   le_ponto(&(r.inf_dir), "segundo");
   printf("\n\nComprimento da diagonal do retangulo: %5.2f\n", diagonal(r));
   printf("\nArea do retangulo: %d", area(r));
   arestas(r);
}
```

```
void le_ponto(Ponto *p, char *s)
{
   int x, y;
   printf("Digite a posicao do %s ponto (x,y): ", s);
   scanf("%d%d", &x, &y);
   p->x = x;
   p->y = y;
}

float diagonal(Retangulo r)
{
   float s1, s2;
   s1 = pow(r.sup_esq.x - r.inf_dir.x, 2);
   s2 = pow(r.sup_esq.y - r.inf_dir.y, 2);
   return sqrt( s1 + s2);
}

int area(Retangulo r)
{
   return ((r.sup_esq.x - r.inf_dir.x) * (r.sup_esq.y - r.inf_dir.y));
}

void arestas(Retangulo r)
{
   printf("\nComprimento das arestas:");
   printf("\nAresta 1: %d", abs(r.sup_esq.x-r.inf_dir.x));
   printf("\nAresta 2: %d", abs(r.sup_esq.y-r.inf_dir.y));
}
```

5) Solução

```
#include <stdio.h>
enum mes { JAN, FEV, MAR, ABR, MAI, JUN, JUL, AGO, SET, OUT, NOV, DEZ };
void main()
{
   enum mes index;
   char *meses[12] = { "Janeiro", "Fevereiro", "Marco",
                       "Abril", "Maio", "Junho",
                       "Julho", "Agosto", "Setembro",
                       "Outubro", "Novembro", "Dezembro" };

   for (index = JAN; index <= DEZ; index++)
      printf("\n%s", meses[index]);
}
```

Cap. 11: ENTRADA/SAÍDA COM ARQUIVOS

QUESTÕES PROPOSTAS

1) Elabore um programa que solicita ao usuário uma string qualquer (fixe um tamanho razoável), bem como o nome de um arquivo. De posse disso, crie um arquivo com o referido nome e escreva neste a string digitada. Antes de fechar o arquivo, salte duas linhas (dentro do arquivo).

2) Modifique o programa anterior a fim solicitar ao usuário uma nova string. Abra o mesmo arquivo e acrescente essa nova string ao final do arquivo. Por fim, exiba na tela o conteúdo do arquivo completo.

3) Elabore um programa que leia o conteúdo de um arquivo de texto qualquer, e calcule a frequência de cada letra (a/A - z/Z) presente no arquivo (desconsiderando a diferença entre MAIÚSCULO/minúsculo). Em seguida, exiba na tela a frequência de cada letra (Ex: a = 25%; b = 4.3%, c = 2.34%; etc). Por fim, informe a letra com maior frequência. Ignore quaisquer outros caracteres. **Observação**: Você está proibido de declarar 26 variáveis diferentes (uma para cada letra)! Encontre uma solução melhor.

4) Admita que você tenha um arquivo texto tal qual o modelo abaixo. Cada linha é composta por um *rótulo*, o caractere '=' e um *valor* cujo tipo depende do rótulo, da seguinte forma: `rótulo=valor`. O rótulo ***animal*** sempre terá um valor do tipo `string`, o rótulo ***inteiro*** um `int` e o ***real*** um `float`. Os "..." indicam que o arquivo pode continuar "indefinidamente" (ele não está presente no arquivo texto, aqui é só um indicativo de que o arquivo não tem apenas esses elementos). Qualquer rótulo pode aparecer em qualquer linha. Logo, não há qualquer ordem pré-estabelecida. Sabendo disso, elabore um programa que aloque 3 vetores dinâmicos com tamanho inicial suportando 10 elementos, para cada um dos tipos acima citados. Abra tal arquivo texto, leia linha por linha e, de posse do rótulo, insira o valor no vetor correspondente. Lembre-se de conferir se o vetor comporta o elemento antes de inseri-lo. Em caso negativo, expanda o vetor em mais 10 posições, a fim de que possa comportar o elemento a ser inserido.

```
animal=gato
inteiro=15
animal=tigre
real=22.5
inteiro=77
...
```

   Finalmente, após ter carregado todo o arquivo, liste na tela os 3 vetores.

5) Desenvolva um programa que solicite ao usuário a quantidade de alunos a ser inserida, aloque um vetor de strings a fim de armazenar cada um dos nomes que serão digitados pelo usuário, um a um. Esse vetor de strings deverá ser passado para uma função que salvará em um arquivo texto cuja primeira linha será a quantidade de nomes e nas demais os nomes dos alunos, propriamente ditos. Confirme para o usuário que o arquivo foi salvo com sucesso.

   OBS: O vetor de strings **não** deverá ser uma matriz (número de colunas iguais para cada linha), ou seja, cada string deverá ser do tamanho exato necessário para armazenar cada um dos nomes. Para tanto, faça o seguinte:

   i. Solicite ao usuário quantos nomes ele deseja digitar. A partir disso, crie um vetor de strings dinamicamente alocado (ou seja, um vetor de ponteiros para char), com este tamanho.

ii. Solicite ao usuário que digite um nome. Armazene-o numa string estaticamente alocada (200 caracteres).

iii. Descubra o tamanho da string digitada no item (ii) e aloque uma string dinâmica com o tamanho exato necessário (atenção!), atribuindo ao respectivo ponteiro criado em (i) e copie a string do vetor estático para o dinâmico.

iv. Repita os procedimentos (ii) e (iii) até que todos os nomes sejam digitados.

6) Desenvolva uma função que faça a leitura de um arquivo de texto com nomes completos de N alunos (criado pela questão anterior), cuja primeira linha do arquivo é o valor de N e nas demais os nomes dos alunos. De posse do valor de N, aloque dinamicamente um vetor de strings e o carregue com os nomes dos alunos lidos a partir do arquivo txt. A função deverá retornar esse vetor de strings dinamicamente alocado. No `main()`, essa função deverá ser chamada e, de posse do seu valor de retorno, imprima na tela todos os nomes lidos. OBS: O vetor de strings, no qual os nomes serão armazenados deverá ser otimizado de maneira análoga a questão 1.

7) Vamos ~~complicar~~ melhorar um pouco mais as questões anteriores! Altere ambas as questões anteriores para lidarem com um vetor de structs (exibido abaixo), ao invés de simples strings. Decida quais serão as alterações necessárias para que o mesmo seja salvo e posteriormente lido do arquivo adequadamente.

```
typedef struct reg {
    char *nome;
    char sexo;
    unsigned char idade;
} Registro;
```

HISTÓRICO DE VERSÕES

[2011-08-03] – v1.1.0

- Removido um exercício inconsistente do Capítulo 8.
- Melhorias nos enunciados dos exercícios 8.1 e 8.2.
- Incluídas mais 4 Questões no Capítulo 11.

[2011-03-16] – v1.0.0

- Adicionadas 3 Questões do Capítulo 11 (Finalmente! ☺). São poucas questões, mas, acredite, darão um certo trabalhinho...

[2010-06-19] – v0.9.5

- Adicionadas Questões Teóricas do Capítulo 6.

[2010-05-16] – v0.9.0

- Adicionados Exercícios Resolvidos/Questões do Capítulo 10.

[2010-05-04] – v0.8.0

- Adicionadas Questões do Capítulo 5.

[2010-04-13] – v0.7.0

- Adicionados Exercícios Resolvidos/Questões do Capítulo 9.

[2010-03-31] – v0.6.0

- Adicionados Exercícios/Questões dos Capítulos 7 e 8.
- Adicionados Exercícios do Capítulo 9.

[2010-03-28] – v0.4.0

- Versão Inicial.